VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Konstrukce kamery s integrovanými funkcemi počítačového vidění

Construction of Camera with Embedded Computer Vision Functions

2018

Petr Kalandra

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Petr Kalandra**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Konstrukce kamery s integrovanými funkcemi počítačového vidění
Construction of Camera with Embedded Computer Vision Functions

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je zkonstruovat a oživit IP kameru schopnou provádět přímou analýzu zaznamenaného obrazu a získané informace dále poskytovat současně s videem. Součástí práce bude softwarová utilita pro konfiguraci konkrétní analytické funkce kamery.

1. Návrh elektroniky kamery.
2. Konstrukce elektroniky kamery.
3. Základní firmware kamery:
a) živý video stream,
b) konfigurační webové rozhraní,
c) programovatelný modul analýzy obrazových dat (např. detekce člověka).
4. Podpůrná utilita pro konfiguraci analytického modulu.
5. Zátěžové testy a zhodnocení výsledků práce.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Ondřej Tomášek**

Konzultant diplomové práce: Ing. Tomáš Fabián, Ph.D.

Datum zadání: 01.09.2017
Datum odevzdání: 30.04.2018

1D16

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 27. dubna 2018 ....

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.

V Ostravě 27. dubna 2018 ....

Rád bych na tomto místě poděkoval kolektivu společnosti CUTTER Systems spol. s r.o., zejména pak Ing. Martinu Řezáčovi a Ing. Ondřeji Tomáškovi, za poskytnutí všech nezbytných prostředků a odbornou pomoc během celého vývoje. Velký dík patří také rodině a přátelům za podporu během let svého studia.

Abstrakt

Cílem této práce je konstrukce funkčního prototypu IP kamery od návrhu hardwaru až po realizaci demonstrační aplikace. V první části přibližuje proces výběru platformy a následně i její integraci do vlastního řešení. Dále je pak krátce přiblížen postup návrhu i výroby DPS a mechanické konstrukce rámu pro prezentaci práce. Následně je pak rozebrána volba analytické funkce, kterou bude kamera vykonávat, jejímž cílem je ukázat možnosti platformy. Jsou popsány i neuronové sítě použité při realizaci aplikace. V poslední části jsou shrnuty práce a odhalené komplikace na jednotlivých částech softwaru výrobku (operačním systému a modulu jádra snímače CMOS, streamovací a analytické aplikaci a webovém rozhraní) a také stručné shrnutí výkonu zařízení.

Klíčová slova: kamera, návrh schématu elektroniky, výroba DPS, konvoluční neuronové sítě, zpracování videa, počítačové vidění, detekce, Tegra K1

Abstract

The objective of this thesis is to construct a functioning IP camera prototype, starting from the hardware design to a demo application. The first part of the thesis focuses on the process of choosing the appropriate platform and subsequently its integration into the project. Furthermore, the process of DPS design and production itself is briefly presented, as well as mechanical construction of the frame in which the device is to be presented. Next, a choice of the analytical function, which the camera will have, is discussed. Its purpose is to show the platform's possibilities. Neural networks used for the application are also described. In the final part the complete works carried out and all the complications and problems are summarized (OS and the CMOS sensor kernel module, streaming and analytical application and web interface). There is also a brief summary concerning the device's computational power.

Key Words: camera, schematic drawing, PCB design, convolutional neural networks, video processing, embedded computer vision, detection, Tegra K1

Obsah

Seznam použitých zkratek a symbolů

ARM – *Advanced RISC Machine*, Architektura procesorů, dnes hojně využívána v mobilních a embedded zařízeních.

CFA – *Color Filter Array*, Mozaiková matice barevných filtrů umístěných před světlocitlivé buňky optického snímače kamery za účelem snížení výrobních nákladů a složitosti technologie výroby.

CMOS – *Complementary Metal–Oxide–Semiconductor*, technologie výroby polovodičů, v kontextu práce je tímto pojmem myšlen optický snímač založený na této technologii.

CSI – *Camera Serial Interface*, standardní rozhraní organizace MIPI pro připojení obrazových snímačů.

CUDA – *Compute Unified Device Architecture*, technologie akcelerace výpočtů s využitím grafických karet vyvíjená společností *NVIDIA*.

DC – *Stejnosměrný elektrický proud.*

DPS – *Deska plošných spojů.*

eMMC – *Embedded MultiMediaCard*, druh nevolatilních pamětí typu flash, de facto paměťová karta v pouzdře určená k osazení přímo na DPS.

FPU – *Floating point unit*, jednotka procesoru řešící výpočty s desetinnou řádovou čárkou.

GFLOPS – *Giga floating point operations*, jednotka udávající výpočetní výkon v miliardách operací nad čísly s plovoucí desetinnou čárkou.

I^{2}C – *Inter-Integrated Circuit*, jednoduchá dvouvodičová komunikační sběrnice pro připojení pomalých komponent s adresací.

IoT – *Internet věcí*, zpravidla drobné senzory se síťovou konektivitou umožňující vzdálený sběr dat.

LDO – *Low-dropout regulator*, stabilizátory s nízkým úbytkem napětí (jsou schopny regulovat na úrovně blízké vstupnímu napětí).

MIPI – *Mobile Industry Procesor Interface Alliance*, organizace s volným členstvím zabývající se standardizací komunikačních rozhraní v mobilních zařízeních.

RFID – *Radio Frequency Identification*, technologie identifikace a lokalizace využívající radiové komunikace na krátkou vzdálenost.

SMBus – *System Management Bus*, přísnější rozšíření sběrnice I^{2}C, využívá se zejména pro konfiguraci periferních zařízení připojených k procesoru.

SoC – *System on Chip*, integrovaný obvod zahrnující procesor a více periferií počítače do jednoho pouzdra.

SOM	–	*System on Module*, modul integrující SoC s dalšími periferiemi, které SoC nezahrnuje v sobě, poskytující tak odrazový můstek pro vývoj zařízení.
SSD	–	*Solid-state drive*, pevný disk bez mechanicky pohyblivých částí, nejčastěji realizovaný nevolatilní pamětí typu flash.
SWD	–	*Serial Wire Debug*, programovací a ladící rozhraní pro mikrokontroléry, jedná se o dvouvodičovou alternativu ke standardu JTAG.

Seznam obrázků

1 Úvod

Počítačové vidění se stále více zapojuje do nejrůznějších odvětví průmyslu. Stále se však jedná o úlohy náročné na výpočetní výkon, a z tohoto důvodu jsou ke spouště dílčích úloh jakož i k samotné analýze obrazu potřeba výkonné periferie. Trendem však je tyto úlohy zjednodušovat a integrovat (alespoň jejich části) přímo do jednotlivých zařízení.

Cílem této práce je především prakticky zkonstruovat kameru schopnou řešit pokročilejší analytické úlohy a zároveň teoreticky přiblížit průběh a úskalí vývoje takového zařízení od počátečních rozhodnutí, přes návrh hardwaru, kompletaci hardwaru, přípravu úloh počítačového vidění až po návrh a tvorbu jednoduché aplikace, která prezentuje možnosti dané platformy na konkrétním analytickém úkolu. Základním předpokladem úspěchu je především demonstrace funkčního konceptu elektroniky s potenciálem rozlišit osobu v obraze za přiměřené rychlosti přenosu videa i detekce pro použití v dohledových aplikacích.

Úvodní kapitoly se zaměřují na důvody, které vedly ke vzniku tohoto projektu a na rozhodnutí, která při plánování projektu musí padnout (zejména kvůli rentabilitě a budoucnosti projektu jako takového). Následuje podrobný popis elektrického zapojení jednotlivých bloků konstruované kamery, stručné shrnutí důvodů jejich integrace, problematika výroby desky plošného spoje a návrhu mechanického řešení demonstračního zařízení. Práce se dále zaobírá typickými úlohami počítačového vidění a přibližuje moderní trend realizace s využitím neuronových sítí, volbu takovéto sítě a její modifikaci a trénování pro konkrétní účel. V posledních kapitolách jsou rozebrána specifika a možnosti vývoje pro zvolenou platformu, která jsou však zpravidla aplikovatelná i na další embedded platformy. Tato zařízení jsou v drtivé většině určena pro použití s operačním systémem Linux. Práce se tedy zaměří i na volbu konkrétní distribuce operačního systému Linux a integraci periferie kamery do jádra systému. V krátkosti jsou popsány různé varianty streamování videa s využitím standardního softwarového vybavení distribucí systému Linux a základní integrace vlastního zpracování a analýzy obrazu do takového softwaru. Ve stručnosti je rozebráno i základní webové rozhraní. Na závěr je zahrnuto i zhodnocení z hlediska výkonu zařízení.

2 Analýza projektu

V následujících podkapitolách je rozebráno, proč a za jakých okolností bylo zvoleno téma práce a jaké požadavky jsou na práci kladeny ze strany společnosti, na jejímž pracovišti je projekt vyvíjen.

2.1 Motivace

Prvotní motivací k vypracování této práce byl návrh společnosti *CUTTER Systems spol. s r.o.*, ve které jsem dlouhodobě zaměstnán, zda by nebylo možné najít oboustranně vhodný průnik mezi diplomovou prací a některým z projektů v rámci pracoviště. Zde ovšem nastal problém, jak zasadit práci z oblasti počítačové grafiky do kontextu firmy.

Společnost se zabývá zejména výrobou produktů pro internet věcí (tzv. IoT), automatizaci, monitoring a telekomunikace. Mezi nejrozsáhlejší řadu výrobků však patří dohledová zařízení pro riziková pracoviště, jako jsou například doly nebo liniové stavby. Většina těchto produktů využívá technologie aktivního RFID. Tato technologie má však svá specifická omezení vyplývající z principu její činnosti. Padl tedy návrh zkonstruovat i prototyp univerzální kamery sloužící k optické analýze, kterou by v případě zájmu zákazníků bylo možné integrovat do portfolia společnosti. Vzhledem k tomu, že se při zběžném průzkumu trhu nepodařilo nalézt žádná zařízení, která by vyhovovala požadavkům firmy, bylo rozhodnuto o volbě tématu práce. Jako její primární využití se předpokládá kontrola vstupu osob do oblastí s omezeným nebo zakázaným přístupem a autonomní upozornění osoby bez nutnosti zásahu nadřazeného systému (lze tedy říci, že jde o včasné varování, nikoli bezpečnostní systém).

V době zahájení práce se na trhu prakticky nenacházel žádný komerční produkt podobného charakteru. Vždy se analýza prováděla zpravidla uvnitř vyšších vrstev systému mimo zařízení, například na serveru nebo ve výpočetním cloudu. Jako jeden z vrcholů lze zmínit například koncem roku 2017 představený čínský systém bezpečnostních kamer [29]. Co se týče analytických funkcí integrovaných přímo v zařízení, zpravidla se jedná o jednodušší dílčí jednoúčelové úlohy, což by měl být i případ našeho produktu. Zde jsou na vedoucí pozici zejména podpůrné systémy řízení v autonomních vozidlech.

2.2 Cíle

Prvním cílem práce bylo tedy navrhnout a vytvořit zařízení, které by bylo zcela v režii společnosti, a to zejména kvůli budoucí integraci do řady jejich produktů, a které by bylo vhodné pro prezentaci. Z tohoto důvodu jsou kromě kamery v hardwaru zahrnuty i další periferie, jako například pomocné komunikační sběrnice. Je kladen důraz na modularitu a rozšiřitelnost hardwarového řešení.

Druhým záměrem práce je vytvořit aplikaci, která umožní přenášet proud obrazových dat z kamery pomocí některého ze standardních multimediálních protokolů a zároveň nad proudem videa provést alespoň základní úpravy obrazu a alespoň přibližně detekovat osobu v obraze (neboť to bude s největší pravděpodobností v nějaké formě primární využití produktu). Opět pokud možno s co největším potenciálem budoucího rozšíření.

Závěr práce si pak klade za cíl shrnout varianty pokračování projektu jako takového a zhodnotit výkonové schopnosti tohoto zařízení a možnosti jeho uvedení do praxe vzhledem k pozorovaným výsledkům.

3 Výběr platformy

Prvním zásadním mezníkem vývoje projektu tohoto charakteru je volba vhodné platformy. Pokusíme se zde tedy stručně nastínit rozhodnutí, která v této fázi vývoje padla a budou popsána specifika zvolené technologie.

3.1 Volba architektury a procesoru

Nejprve bylo nutné zvolit obvod, který bude analýzu provádět. Zde jsou k dispozici prakticky tři možnosti a to:

- digitální signálové procesory (DSP),
- programovatelná hradlová pole (FPGA),
- aplikační procesory (CPU).

Digitální signálové procesory jsou velmi specifická zařízení, která vyžadují důkladnou optimalizaci a znalost architektury. Bohužel s nimi nemám mnoho zkušeností a ani dostatečné vývojové prostředky. To by se odrazilo negativně na ceně a délce celého vývoje. Řešení bylo tedy zamítnuto hned v počátku. Zmiňuji jej ale z toho důvodu, že krátce před započetím práce na projektu uvedla společnost *Cadence Design Systems, Inc.* na trh první tzv. IP core zaměřený na provoz neuronových sítí na architektuře DSP [8].

Hradlová pole jsou podobný případ jako digitální signálové procesory. V případě jejich použití by aplikace nebyla příliš univerzální a každá vytrénovaná neuronová síť (nebo jiný algoritmus) by se musela efektivně překonvertovat do jazyka VHDL a optimalizovat pro konkrétní čip. Existuje zde sice již celá řada nástrojů s tímto účelem, ale stále se jedná o „mravenčí“ práci a v časovém horizontu realizace práce by toto řešení nebylo pravděpodobně zvládnutelné.

Zejména z časových důvodu, na základě zkušeností a s přihlédnutím k univerzalitě řešení jsme zvolili využít aplikačního procesoru. Většina embedded zařízení je dnes založena na architektuře ARM. V případě aplikačních procesorů je de facto standardem přítomnost FPU. Jádra však nedosahují zpravidla takového výpočetního výkonu. Proto bylo voleno z procesorů, které mají integrovaný grafický akcelerátor. Těch je na trhu celá řada, ale v době zahájení prací bylo na trhu jen minimum embedded procesorů zahrnujících i tzv. compute capability, která by byla pro realizaci složitějších algoritmů a neuronových sítí nejvhodnější. Na základě množství (i amatérských) projektů, dostupnosti vývojových modulů a částečně také na základě marketingu výrobce došlo na volbu procesoru z řady *Tegra*.

3.2 Procesory řady Tegra

Procesory řady *Tegra* jsou vyvíjeny americkou společností *NVIDIA Corporation*, která je vnímána jako společnost s vedoucí pozicí na trhu v oblasti počítačové grafiky a zpracování obrazu. Jedná se o takzvané *System On Chip* řešení, což znamená, že v rámci jednoho pouzdra je integrována většina periferií běžného počítače, jako jsou akcelerátory grafiky, zvuku, komprese videa a dalších. V současnosti jsou k dispozici řady *Tegra K1*, *Tegra X1* a *Tegra X2*. Přibližně na začátku druhého kvartálu roku 2018 by na trh měla být uvedena zcela nová řada s kódovým označením *Xavier* [10]. Jedná se o vícejádrové procesory s jádry ARM řady Cortex-A s přídavným grafickým multiprocesorem s podporou výpočetní technologie CUDA, která bude dále rozebrána v jedné z následující částí práce.

Ačkoliv by z hlediska výkonu bylo nejoptimálnější použití novějších modelů *X2*, museli jsme se při výběru rozhodovat zejména podle dostupnosti na trhu a ceny modulu. Procesory jsou pro menší výrobce prakticky nedostupné a navíc návrh kompletního zapojení není finančně výhodný, protože značně natahuje dobu vývoje a je náročné zapojení odladit, zejména co se týče návrhu DPS. Z tohoto důvodu jsou v poslední době mezi menšími výrobci elektroniky a amatérskou veřejností preferována řešení v podobě *SOM*, které rozeberu v následující podkapitole.

Bohužel pro varianty *Tegra X1* a *Tegra X2* nebylo v době výběru příliš dostupných modulů (prakticky jen oficiální platforma *Jetson*, a to se do po dobu tvorby práce příliš nezměnilo. Naopak je zřejmé, že se výrobce snaží prodávat pouze své moduly a ostatní výrobci se tak soustředí spíše na výrobu doplňků k oficiálnímu vývojovému kitu. Oficiální moduly jsou však cenově posazeny velmi vysoko. Model se slabší *Tegrou X1* se pohybuje cenově okolo 350$, silnější varianta s *Tegrou X2* pak okolo 500$. Z tohoto důvodu byl po dohodě s vedoucím práce zvolen model *Tegra K1*, který se v modulech pohyboval na cenové hladině okolo 200$[1], což bylo akceptovatelné.

3.3 Výběr SOM

Následoval výběr konkrétního modulu. Výrobců SOM s procesory K1 je na trhu celá řada, zejména od oficiálních partnerů společnosti *NVIDIA*, jejichž seznam je k dispozici na webu společnosti [9]. Z modulů dostupných od zde uvedených výrobců byl vybrán SOM *Apalis TK1* od švýcarské společnosti *Toradex*.

Hlavním argumentem pro volbu tohoto modulu bylo, že výrobce poskytuje i další moduly z řady Apalis s jinými procesory (konkrétně slabší *Tegra T30* a procesory *iMX 6* společnosti *NXP*). Moduly mají standardizované rozhraní a společnost se snaží i o kompatibilitu specifických (nestandardních) pinů modulu. V budoucnu by to tedy mělo přinést minimalizaci úprav elektroniky a operačního systému. Navíc je řada modulů *Apalis* neustále doplňovaná a je tak pravděpodobné, že se časem objeví i další modul s novějším procesorem řady *Tegra* (to ovšem závisí na výrobci procesorů, ale společnost *Toradex* o to již dlouhodobě usiluje [11].). Moduly také nabízí vysokou míru integrace periferií.

---

[1] Ceny převzaty z amerického internetového obchodu s komponentami *Arrow Electronics*.

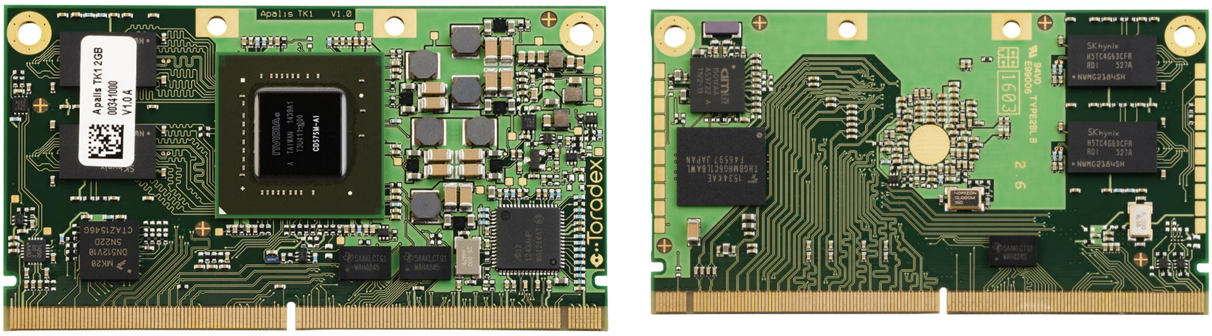
Druhým důvodem pro tento výběr bylo rozsáhlé a snadno dostupné množství dokumentace jak v podobě „wiki“ článků, tak v podobě PDF dokumentů a příruček. Tato dokumentace byla při vývojových pracích (zejména elektroniky) do značné míry využívána. Výhodnou se jevila také rychle reagující komunitní i oficiální podpora na diskuzním fóru výrobce. Poslední výhodou pak je, že se jedná o evropského výrobce s vlastním e-shopem a je tedy možné veškeré moduly přímo objednat, navíc bez nutnosti kontaktu s obchodním oddělením. Společnost taktéž slibuje dlouhodobou podporu a dostupnost [12].

3.4 Tegra K1 a SOM Apalis TK1

Aby mohl čtenář zvolenou platformu srovnat s jinými, je na tomto místě vhodné stručně shrnout základní rysy procesoru i SOM.

Tegra K1 sestává z jader založených na architektuře ARM Cortex-A15 r3 označovaných jako *NVIDIA 4-Plus-1*^{TM} *Quad-Core*. Technologie sestává ze čtyř jader optimalizovaných pro výkon (tzv. blok G - General) a jednoho jádra optimalizovaného pro spotřebu (tzv. blok LP - Low Power) [13]. Jádra mohou být taktována až na frekvenci 2,3 GHz, ale zpravidla se využívá frekvencí nižších pro úsporu energie, jestliže jádra nejsou zcela vytížena. Jedná se o 32-bitovou architekturu. Dále je zahrnut jeden grafický multiprocesor s podporou technologií *CUDA 6.5*, *OpenGL 4.5*, *OpenGL ES 3.1* a *Vulkan*. Ten je založen na mikroarchitektuře Kepler a sestává ze 192 CUDA jader pracujících na frekvencích od 756 do 951 MHz. Udávaný výkon grafického multiprocesoru je 290 až 365 GFLOPS při výpočtech s FP32. K procesoru je možné připojit až 8GB operační paměti typu DDR3L nebo LPDDR3 s udávanou propustností 17GB/s. Dále je integrován signálový procesor, který je využíván zejména pro kompresi videa knihovnou *OpenMAX*. Ten je však bohužel využitelný prostřednictvím podpůrných knihoven a aplikací od výrobce a není jej možné využít uživatelsky jiným způsobem. Procesor je vyrobený 28 nm technologií. Udávaná spotřeba činí 8 W. Informace o parametrech převzaty z [14].

SOM *Apalis TK1* integruje procesor do modulu, který nabízí další periferie, ale omezuje maximální frekvenci CPU na 2,1 GHz. Modul zahrnuje operační paměť o kapacitě 2 GB typu DDR3L a nevolatilní flash paměť typu eMMC pro uložení operačního systému a aplikací o kapacitě 16 GB. Dále je integrován řadič fyzické vrstvy sběrnice Ethernet 1000Base-TX model *I210* od společnosti *Intel*. Z důvodu zajištění kompatibility se standardním konektorem modulů Apalis byl přidán pomocný mikroprocesor *MK20* od společnosti *NXP* o maximální frekvenci 100 MHz, který poskytuje transparentní rozhraní ke sběrnicím, o které hlavní procesor rozšiřuje, pomocí modulu jádra poskytovaného v distribuci operačního systému. Je možné jej však využít i jiným způsobem. Dále jsou integrovány pomocné systémové periferie jako jsou obvod reálného času, zdroj hlavního hodinového signálu CPU, správu napájení a analogové audio (zajištěno čipem SGTL5000). Modul se připojuje pomocí konektoru *MXM3* (Mobile PCI Express Module), který je standardem pro mobilní grafické karty v laptopech [15]. Tento konektor je také ideální z hlediska osazení v menších podnicích, lze jej totiž osadit ručně, neboť má piny pouze po obvodu slotu. I to byl jeden z faktorů hovořících pro využití tohoto modulu. Informace o vybavení



(a) Pohled z horní strany (b) Pohled z dolní strany

Obrázek 1: Fyzická podoba modulu Apalis TK1.

SOM jsou převzaty z oficiální dokumentace modulu [5]. Blokové schéma modulu je přiloženo na straně 11 citované dokumentace. Pro názornost je na obrázku 1 zachycena fyzická podoba modulu. Obrázek převzat z [16].

4 Návrh elektroniky kamery

Tato kapitola si klade za cíl co nejstručněji přiblížit volbu jednotlivých komponent a řešení, poukázat na některá zajímavá řešení či na případné potenciální překážky. Veškerá zde probíraná schémata jsou součástí tištěné přílohy a společně s návrhy desek plošných spojů i ve formě příloh elektronických. Výhodou elektronické verze je snadnější využití v případě navazujících prací s využitím identického návrhového softwaru.

4.1 Rozdělení elektroniky

Vzhledem k požadavkům na integraci kamery do portfolia firmy, bylo vhodné zajistit, aby zařízení podporovalo komunikaci po sběrnicích podporovaných v ostatních produktech. Nebylo však vhodné zabývat se v časovém horizontu práce řešením konkrétní integrace, neboť se budou požadavky lišit podle případných odběratelů. Dále pak podle světelných podmínek nebo specifických faktorů cílového umístění produktu může být vhodné změnit nebo doplnit způsob snímání (například volbou jiného optického snímače, termokamerou, rozšířením na stereoskopickou kameru, time-of-flight snímačem, ap.). Ačkoliv bude zřejmě v takovéto situaci nutné přepracovat zapojení a desky plošného spoje, dokáže přichystané řešení značně zkrátit dobu vývoje a usnadnit vývojové práce, a tím i snížit náklady. Z tohoto důvodu bylo zvoleno modulární řešení, a to konkrétně rozdělení na:

- Hlavní desku (tzv. *carrier board*),
- Moduly sběrnice Ethernet (konkrétně 1000BASE-TX a 100BASE-FX),
- Modul čelního panelu s kamerou,
- Modul I/O (ten však nebyl v rámci práce řešen, pouze byla připravena hlavní deska pro jeho připojení).

4.2 Volba návrhové aplikace

V rámci společnosti se v současné době používá k návrhu elektronických schémat a desek plošných spojů návrhový software *Eagle*. Jedná se o poměrně jednoduchou aplikaci, takže si designer musí většinu pokročilých záležitostí vyřešit sám. Jedná se zejména o absenci nástrojů pro routování sběrnic nebo přizpůsobení délek signálů a impedancí. Ačkoliv by bylo jednodušší použít některý z profesionálnějších produktů (např. *Altium*), které zpravidla svými nástroji usnadňují routing vysokofrekvenčních spojů, rozhodl jsem se použit nástroj, který byl v daný okamžik k dispozici.

Použita je konkrétně verze *Eagle 7.7.0* ve variantě *Ultimate* bez autorouteru. Použití komerční verze programu bylo nezbytné vzhledem k faktu, že se do budoucna jedná o komerční projekt, ale také předpokladu, že budou přesaženy limity bezplatné verze a to jak rozměry desky, tak počtem vrstev desky. Tento předpoklad se nakonec ukázal jako opodstatněný. Jako výhoda této volby se ukázala možnost rozdělit elektrické schéma do více listů a mít jej tak logicky členěné podle bloků. Návrhy vytvořené v komerční verzi aplikace lze pro prohlížení a kopírování otevřít i v bezplatných verzích, nelze je však modifikovat.

Další z výhod programu *Eagle* je také vzájemná provázanost editoru schémat a desky plošných spojů, což nebývá samozřejmostí ani u dražších aplikací. Veškeré změny v editoru schémat se okamžitě projeví i na desce.

4.3 Návrh hlavní desky

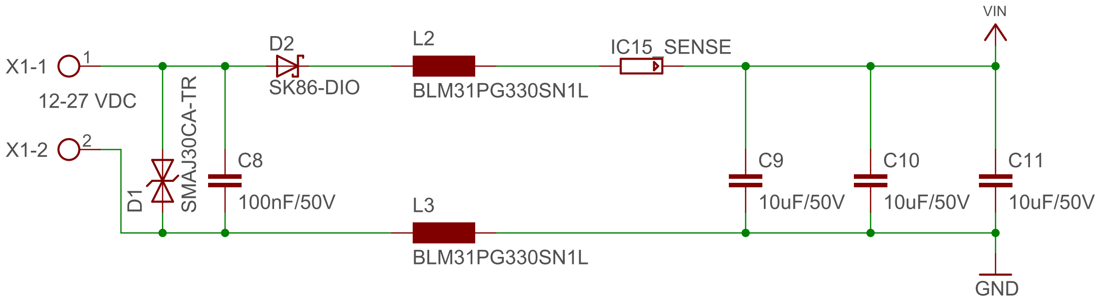
Jednou ze zásadních vlastností modulu *Apalis TK1* byla i rozsáhlá dokumentace, která značně zjednodušila návrh hlavní desky. Při návrhu byla využívána zejména oficiální příručka *Apalis Carrier Board Design Guide* [6] a vycházelo se zejména z referenčních zapojení. I přes tento fakt se ovšem doba strávená jen nad tvorbou schématu hlavní desky překročila 100 hodin, neboť ne vždy bylo zapojení ideální nebo jej bylo potřeba přizpůsobit pro konkrétní účel. Aby nedocházelo k neúměrnému navyšování délky tohoto textu, budou schémata v jednotlivých sekcích pro názornost umístěna jen na místech, kde se použité zapojení zásadně liší od referenčního zapojení z příručky nebo u schémat, která jsou původní či adaptovaná z jiných zdrojů, aby bylo možné snadněji objasnit princip činnosti.

4.3.1 Napájení hlavní desky

Pro kompatibilitu s většinou výrobků společnosti byl zvolen rozsah stejnosměrného napájecího napětí 18-27 V. Připojení přívodních napájecích kabelů je řešeno šroubovací svorkovnicí, protože se jedná o nejuniverzálnější řešení hojně využívané i na komerčních průmyslových výrobcích. Jako ochrany proti přepětí na vstupu je využito bipolárního 30 V transilu *SMAJ30CA-TR*, který zajistí snížení napětí pod tuto mez. Dále je za kladnou napájecí svorku přidána schottkyho dioda *SK86-DIO* pro zajištění ochrany proti přepólování. Zapojení vstupní napájecí části je možno shlédnout na obrázku 2.

Samotný napájecí zdroj je zapojený podle referenčního zapojení. Jedná se o integrovaný obvod *TPS51120RHB*. Ten poskytuje pomocná stejnosměrná napájecí napětí 5 V a 3,3 V se zatížitelnosti do 100 mA, která jsou dále využívána pro dva samostatné výkonové bloky spínaných zdrojů vytvářejících identická napětí do maximálního proudu přibližně 6 A. Zapojení také zajišťuje korektní sekvenci spínání napájecích napětí, jejichž pořadí je nezbytné dodržet pro správnou funkci komponent. Také se tím minimalizovalo riziko poškození SOM.

Pětivoltová větev je využita pouze pro napájení Ethernetového modulu a modulu I/O. Do budoucna je možné, že tato větev bude zredukována kvůli nedostatečnému využití. Pro potřeby práce však zůstala zejména z důvodu využití referenčního zapojení, jelikož tvorba výkonných spínaných zdrojů napětí s sebou přináší různá úskalí (např. v podobě elektromagnetického rušení). Bylo tedy výhodné zůstat prozatím u ověřeného zapojení.



Obrázek 2: Schéma zapojení napájecího vstupu

Větev 3,3 V pak slouží k napájení všech hlavních komponent, zejména SOM. Pro periferie připojené k procesoru je pak dále spínáno výkonovým tranzistorem *DMP2022LSS-3*, který zajistí jejich připojení až po inicializaci modulu. Tranzistor je řízen nepřímo modulem přes spínání 5 V větve.

4.3.2 Rozhraní modulu Mini PCIe

Aby byla zajištěna případná možnost bezdrátové konektivity (WiFi, GSM, ap.) nebo dalšího rozšíření, byl do zařízení zahrnut slot pro modul *Mini PCI Express*. Zde je využito referenčního zapojení. Ke slotu jsou připojeny sběrnice *PCIe x1*, *USB 2.0* a konfigurační sběrnice *SMBus*. Dále je připojen slot na SIM kartu, pro případ, kdy by bylo využito modemu. Dle standardu vyžaduje slot pomocné napájecí napětí 1,5 V. Je zde tedy zahrnut i napájecí lineární zdroj LDO *TPS74801DRCT*.

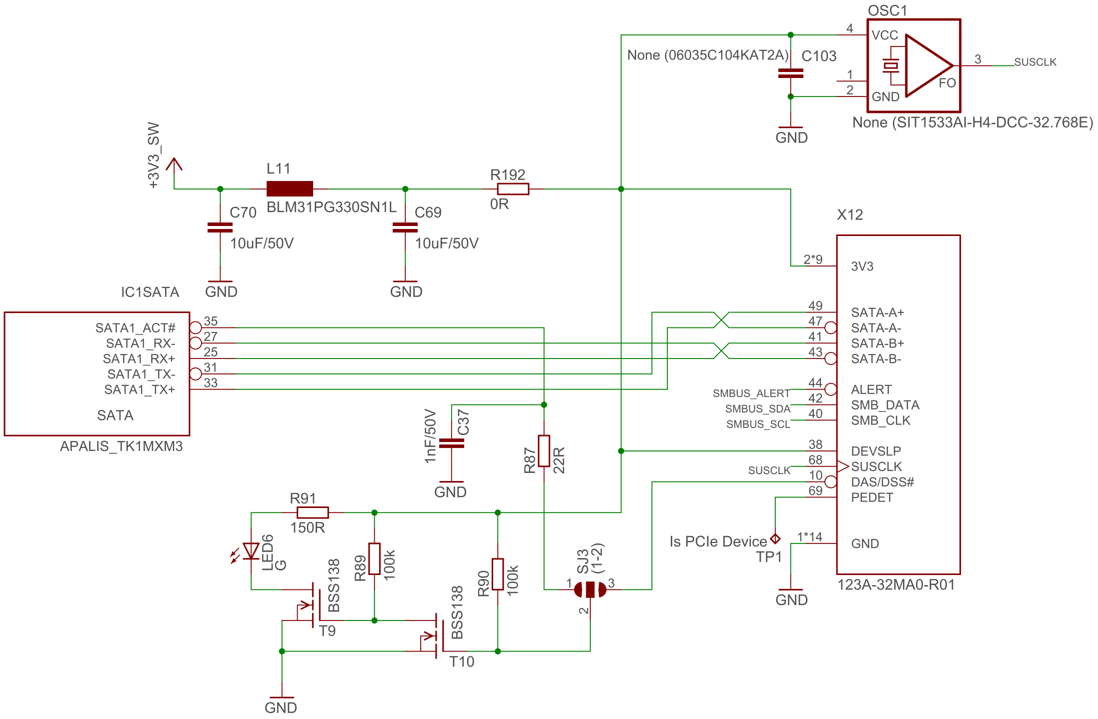
4.3.3 Rozhraní modulu M.2

Za účelem zvětšení úložného prostoru pro případný záznam či jako úložiště při vývoji byl do návrhu zařazen i slot pro SSD disky *M.2*. Moduly (i sloty) *M.2* existují v několika variantách vzájemně rozlišených jednopísmenným označením [17]. Na pevných discích typu SSD je nejčastěji využíván klíč typu *B+M* (tedy lze je umístit do slotů typu *B* i typu *M*). Důvod pro tuto kombinaci je ten, že se v obou případech na shodných pozicích nachází kontakty pro sběrnice *SATA* a *PCIe x4*. Na desce se slotem je pak zvolen klíč podle poskytovaných rozhraní.

Jelikož zbylo k dispozici jen nevyužité rozhraní *SATA*, byl zvolen slot s klíčem *M*. Při návrhu zapojení však nastal problém s přístupem k oficiálnímu standardu konsorcia PCI-SIG. Pro bezplatný přístup k dokumentům je nezbytné členství v organizaci, ostatním je celý standard zpřístupněn v tištěné podobě za cenu 2000$ [18][2]. Jelikož tato částka byla mimo rozpočet projektu, a navíc nebylo třeba znát celou specifikaci, ale jen nezbytnosti pro připojení disku na sběrnici *SATA*, bylo využito nápadu pokusit se informace nalézt přímo v dokumentaci některého disku. Na základě předchozích zkušeností jsem sáhl po dokumentaci disků společnosti *Intel*. Tyto disky mají ve většině případů hlavní výhodu oproti konkurenci v relativně nízké spotřebě,

---

[2] Dokument NR48 - PCI Express M.2 Specification (hard copy)



Obrázek 3: Schéma zapojení slotu M.2

což je na takovémto druhu zařízení výhodou. Jako první byla objevena dokumentace disků řady *540s*. Ta obsahovala celý výpis elektrických signálů vyvedených na rozhraní modulu [19]. Další detaily již pak nebylo obtížné dohledat. Výsledné zapojení je na obrázku 3.

K modulu je tedy přivedeno rozhraní *SATA*, dále pak konfigurační sběrnice *SMBus*. Signál *DEVSLP* je trvale připojen k napájecímu napětí, čímž je zabráněno přepnutí modulu do režimu spánku. Ten však není třeba, režim spánku není předpokládán ani v případě SOM. Z tohoto důvodu také není přítomen oscilátor *OSC1* 32,768 kHz, který by modulu poskytoval hodinový signál v tomto režimu na pinu *SUSCLK*. Signál *DAS/DSS#* slouží jako signalizace aktivity disku. Obvod v levém dolním rohu pak slouží ke spínání LED diody *LED6*. Pájecí propojkou *SJ3* je pak umožněna volba zdroje blikání a to buď z disku SSD (propoj 2-3), nebo z procesoru (propoj 1-2), anebo jej zakázat úplně. Napájení je shodné s napájením SOM a je přidán filtrační ferit *L11* v kombinaci s kondenzátory *C69* a *C70*.

4.3.4 Rozhraní modulu kamery

Rozhraní modulu kamery je z hlediska zapojení relativně triviální. Propojení hlavní desky s kamerovým modulem je realizováno plochým flexibilním kabelem, zejména kvůli úspoře místa. Jedná se o přímé propojení dvou sběrnic *MIPI CSI* s konektorem pro kamerový modul. Každá ze sběrnic má připojeny čtyři samostatné datové páry a jeden hodinový pár (jedná se v obou případech o diferenciální vedení, standard *D-PHY*).

Dále je připojena dedikovaná sběrnice I^2C pro konfiguraci čipů CMOS (a dalších komponent na kamerovém modulu), neboť rozhraní *CSI* umožňuje pouze jednosměrnou komunikaci ze snímače do procesoru.

Aby bylo možné rozšířit funkcionalitu kamerového modulu (zejména v případě potřeby dalších řídících signálů), předpokládá se využití mikrokontroléru. Z tohoto důvodu je na konektor přivedeno programovací rozhraní *SWD*, které slouží pro případnou aktualizaci firmwaru v mikrokontroléru. Řešení je zmíněno v kapitole 4.3.8, kde se nachází shodné rozhraní pro modul I/O. V případě, že by bylo nezbytné zaznamenávat i zvuk, je vyveden i analogový vstupní signál pro připojení mikrofonu.

Kamerový modul je napájen ze samostatného 12 V zdroje, který je realizován obvodem *LM22676MR-ADJ*. Důvodem je potenciálně vyšší spotřeba kamerových snímačů (např. při realizaci modulu s dvojicí kamer) nebo dalších komponent (výhřev optiky, přísvit a další).

4.3.5 Rozhraní vyměnitelného modulu sítě Ethernet

Jelikož v sobě SOM integruje i fyzickou vrstvu pro *1000BASE-TX Ethernet*, je realizována nejjednodušší varianta. Tou je vyvedení signálů všech komunikačních párů a pomocných signálů (signalizační LED a reference) na konektor vyměnitelného modulu.

Na konektoru modulu je dále poskytnuto konfigurační rozhraní *SMBus* a napájení 5 V a 3,3 V (to slouží především jako pomocné, maximální zatížitelnost pinu konektoru je 200 mA) z hlavního zdroje.

Pro připojení vyměnitelného modulu byl použit 32-pinový konektor výrobce *Harting* umožňující jeho přímé bezkabelové propojení s hlavní deskou.

4.3.6 Rozhraní HDMI

Rozhraní HDMI je zpřístupněno na standardním konektoru. Lze jej tedy připojit klasickým kabelem k monitoru.

Referenčního zapojení z „Design guide“ zde není využito, protože se ukázalo být příliš komplikovaným. Namísto něj bylo s výhodou použito obvodu *TPD12S016PWR*, který vyžaduje jen tři externí 100 nF filtrační kondenzátory na napájecích pinech. Tím se celé zapojení zjednodušilo na šest komponent včetně konektoru a byla tak zajištěna jak ESD ochrana signálů vedoucích z konektoru, tak převod úrovní všech pomocných signálů (HDMI využívá úrovně 5 V, SOM vyžaduje 3,3 V). Konkrétně se jedná o signály *Hot-plug detekce*, *HDMI-CEC* a *DDC* (konfigurační kanál využívající dedikované rozhraní I^2C). Výhodou je také zjednodušené rozvádění diferenciálních signálů. Obvod je navržený tak, že v případě správného umístění poblíž konektoru lze tyto signály vézt „skrz“ pin a návrhář tak nemusí řešit křížení nebo značnou změnu impedance vedení.

Pro odrušení je přidán ferit mezi kostru konektoru HDMI a signálové zemnící piny.

V případě komerčního nasazení výrobku je použití rozhraní HDMI zatíženo licenčními poplatky. Bude-li nezbytné redukovat cenu výrobku, je možné rozhraní odstranit či nahradit jinou alternativou nabízenou modulem. K dispozici je například rozhraní *Embedded DisplayPort*. Potenciálním řešením problému může být i náhrada HDMI konektoru za konektor *DVI*.

4.3.7 Rozhraní USB

Dalším poskytnutým rozhraním je *USB* v režimu *Device*. Zapojení zcela vychází z referenčního a zahrnuje ochranu proti přepětí i filtrační ferity. Vyvedeno je prostřednictvím samice standardního konektoru *USB-B Mini*. Zahrnuto bylo zejména kvůli možnosti streamování v případě absence *Ethernetu* či pro aktualizaci systému prostřednictvím zavaděče.

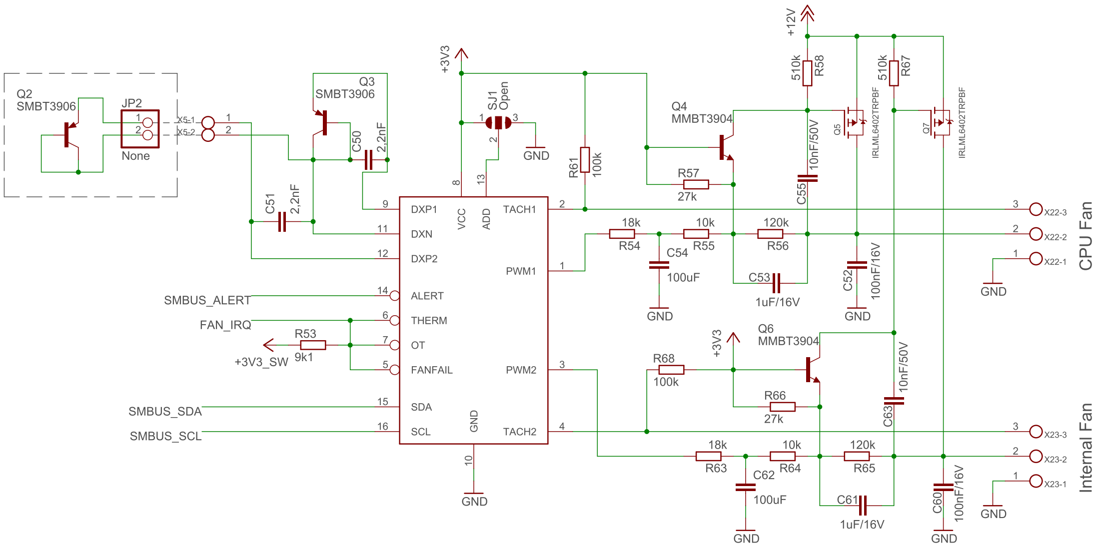
4.3.8 Rozhraní I/O modulu

Jak již bylo zmíněno dříve v úvodu této kapitoly (4.1), byla deska navržena tak, aby k ní mohl být připojen rozšiřující modul se standardními sběrnicemi používanými v mikroelektronice a automatizaci. Konkrétně se jedná o:

- *Audio* (monofonní analogový vstup i výstup) pro poskytování akustické signalizace nebo záznam zvuku, případně pro kontinuální měření analogových veličin,
- *CAN* pro komunikaci na delší vzdálenost (avšak bez řadiče fyzické vrstvy jen v podobě signálů *Receive* a *Transmit*),
- 2x *PWM* (pulzně šířková modulace) pro kontinuální regulaci,
- 4-bitové *SDIO* (včetně detekce vložení karty) pro rozšiřující zařízení vyžadující rychlou komunikaci či paměťovou kartu SD,
- *SMBus* pro pomalou komunikaci s obvody modulu,
- *S/PDIF* (digitální audio) pro případný (avšak nepředpokládaný) přenos audia na delší vzdálenosti,
- *SWD* pro aktualizaci firmwaru v případném mikrokontroléru na modulu,
- *UART* (jedenkrát včetně řídících signálů a dvakrát redukovaný na příjem a vysílání) pro asynchronní komunikaci s periferiemi.

Modulu je poskytnuto přímo neregulované vstupní napájecí napětí, aby bylo využití co nejuniverzálnější, a také aby bylo možné po kabelu přenést co největší výkon. Dále pak napětí 5 V (maximální zatížení 800 mA) a napětí 3,3 V sloužící opět spíše jako signalizační (maximální zatížení 200 mA).

Propojení je realizováno konektory pro plochý flexibilní kabel, který je shodný s kabelem kamerového modulu.

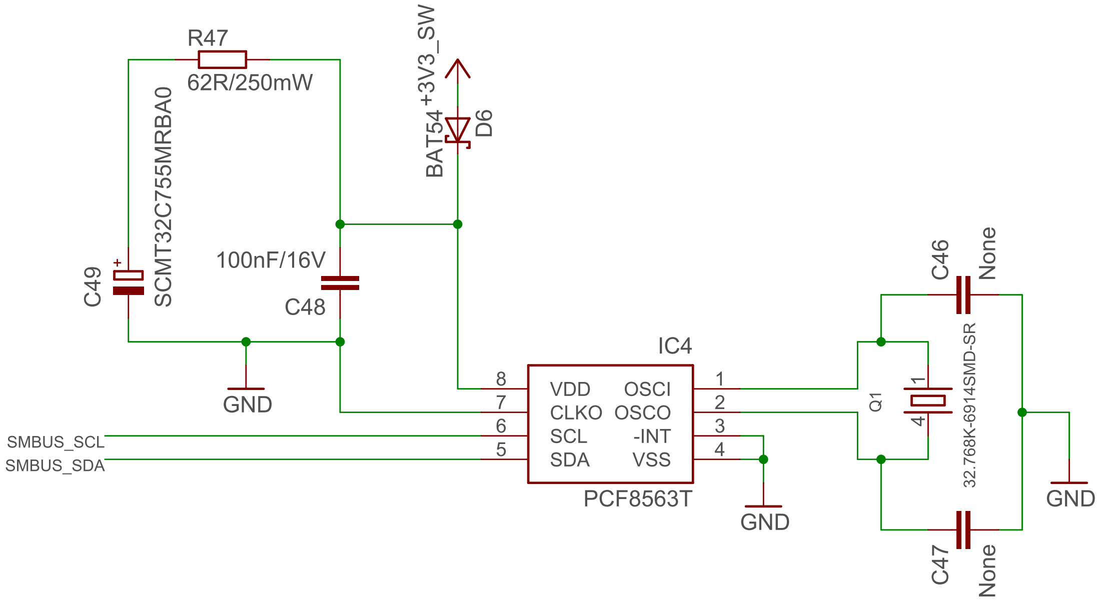


Obrázek 4: Schéma zapojení regulátoru otáček ventilátorů

Také bylo třeba zahrnout realizaci rozhraní *SWD*, která není ze SOM poskytována. Signálově je podobné sběrnici *SPI*. Jako výchozího zdroje vývoje bylo využito zapojení použitého na vývojových kitech mikrokontrolérů *Kinetis*, kde je tento přístup s úspěchem aplikován [20]. Z pohledu procesoru se jedná o klasické *SPI* s dvěma přidanými řídícími signály. Těmi jsou řízeny oddělovací buffery, které slouží ke správné konverzi na *SWD*. Správnou obsluhu rozhraní je pak nutné zajistit na straně softwaru. Avšak nejedná se o problém, neboť je na sběrnici definovaný komunikační protokol pro konkrétní obsluhu mikrokontroléru. Z pohledu softwaru by se tak pouze jednalo o přidanou vrstvu mezi protokol a řadič sběrnice.

4.3.9 Chlazení

Procesor má příkon až 8 W, celý modul pak pochopitelně o něco vyšší. Je tedy nezbytné zajistit adekvátní chlazení. Za tímto účelem je na desce pod procesorem umístěn PNP tranzistor *SMBT3906* jehož přechod emitor-báze je využit jako snímač teploty. Další může být volitelně připojen na dvouvývodovém konektoru v sousedství těch pro připojení ventilátorů. Snímače jsou připojeny k dvoukanálovému řídícímu obvodu ventilátorů *MAX6639*. Ten má však výstupy regulované pulzně šířkovou modulací, se kterou mohou mít některé typy ventilátorů problém. Proto je výstup zapojený podobně jako v aplikační poznámce číslo 3530 výrobce obvodu [21]. Tím by měla být zajištěna optimální regulace a snížení hladiny hluku. Obvod lze řídit sběrnicí *SMBus* a poskytuje zpět do procesoru signál přerušení pro případ poruchy. Ve výchozím stavu obvod reguluje na 50% výkonu. Ventilátory jsou napájeny ze samostatného zdroje 12 V, který je realizován obvodem *LM2841XMK-ADJL*. schéma zapojení regulátoru je k dispozici na obrázku 4. Primární ventilátor je uchycen čtveřicí šroubů M2.5 do závitů v originálním chladiči modulu.



Obrázek 5: Schéma zapojení obvodu hodin reálného času

4.3.10 Obvod hodin reálného času

Vhodným doplňkem je také obvod hodin reálného času, který lze využít například pro časové označení videa nebo pojmenování uloženého záznamu. SOM sice integruje jeden takový obvod, avšak v „Design guide“ je doporučeno přidat vlastní, neboť ten na modulu má vyšší spotřebu energie a nelze jej dlouhou dobu zálohovat při výpadku napájení. Z tohoto důvodu byl při návrhu zahrnut i obvod *PCF8563T*, který je výhodný zejména kvůli nízké ceně, nízké spotřebě a jednoduchosti zapojení. K němu může být volitelně připojen vysokokapacitní kondenzátor (tzv. Supercapacitor). Ve schématu je uveden kondenzátor *SCMT32C755MRBA0* s kapacitou 7,5 F, který by měl umožnit zálohování na více než rok. To je samozřejmě velmi maximalistická varianta (realita bude pravděpodobně kratší, výpočet vycházel jen z katalogových hodnot) a tak dlouhá doba nebude nikdy využita, navíc je tato varianta velmi finančně nákladná. Je tedy možné osadit libovolný kondenzátor (s povoleným napětím 3 V a více) podle aktuální potřeby. Oscilační obvod je možné doladit dvěma kondenzátory po stranách krystalu. Připojení k procesoru je realizováno sběrnicí *SMBus*. Celé zapojení je ukázáno na obrázku 5.

4.3.11 Monitorování proudu

Ve finální fázi vývoje je dobré mít možnost získat přehled o spotřebě produktu. Za tímto účelem byly na hlavní napájecí větve (vstupní neregulované napětí, 5 V a 3,3 V) přidány obvody měřící proud s využitím hallova jevu *ACS711LC*. Jedná se o obvod, který má na jedné straně pouzdra dvě dvojice pinů, přes které protéká měřený proud a na druhé straně se nachází napájení, signalizace nadproudu a analogový výstup. Analogový signál je pak dále přímo připojen na analogové vstupy procesorového modulu.

Maximální měřitelný proud je 12,5 A a je možné rozeznat i směr toku proudu - při nulovém průtoku je na výstupu napětí o hodnotě poloviny napájecího napětí, podle směru toku proudu se pak vychyluje nad nebo pod tuto úroveň.

4.4 Moduly sítě Ethernet

Jelikož je zařízení koncipováno primárně jako IP kamera, dá se předpokládat, že Ethernet bude v zapojení esenciální. Protože ale řada produktů firmy CUTTER Systems spol s r.o. využívá optickou 100 MBit/s konektivitu, byly vytvořeny dva moduly, které lze do hlavní desky zasunout.

Prvním z dvojice je modul *1000BASE-TX*. Jedná se o rozhraní poskytované přímo z SOM, je proto svou konstrukcí velmi jednoduchý. Obsahuje konektor RJ-45 s integrovanými transformátory *L829-1J1T-43*, na který jsou přímo přivedeny signály sběrnice *Ethernet* a signalizačních LED. Dále pak jen několik málo pasivních součástek zejména pro filtraci rušení z kabelu a kostry konektoru.

Druhým je pak komplikovanější modul optické konektivity *100BASE-FX*. Ten je postaven okolo katalogového zapojení vývojového kitu centrálního obvodu *KSZ8863FLL*[3]. Jedná se o čip realizující funkci switche, zde sloužící jako převodník média (fyzické vrstvy). Z něj je na stranu optického rozhraní připojen přes kapacitní vazbu a odporové děliče (pro zajištění korektních napěťových úrovní) integrovaný optický duplexní transceiver *SC1315-40APO*. Na druhé rozhraní jsou pak připojeny páry 100 MBit/s metalické vrstvy, které jsou přes oddělovací transformátor a konektor modulu připojeny k fyzické vrstvě SOM. Pravděpodobně by mělo být možné realizovat toto propojení i bez oddělovacího transformátoru například kapacitní vazbou, ale toto řešení je riskantní z hlediska spolehlivosti a liší se také podle použitých obvodů. Zapojení by tak nebylo možné převzít do jiných projektů bez nutnosti dalšího ladění. Nejvíce času bylo u tohoto modulu věnováno správnému zapojení „Strap pinů“. Ty slouží ke stanovení výchozí konfigurace. Díky tomu je možné jej používat jako media konvertor přímo bez nutnosti dalšího nastavování prostřednictvím *SMBus* (i přes to je ale zapojena pro poskytnutí možnosti nastavení obvodu změnit, případně rozeznat přítomnost optického modulu).

4.5 Návrh kamerového modulu

Návrh kamerového modulu sestával z výběru snímače, jeho integrace do schematu a univerzálního návrhu rozložení komponent mezi více DPS.

4.5.1 Volba snímače

Jednou z nejdůležitějších částí vývoje celého zařízení byla bezpochyby volba senzoru. Tento bod však zabral delší dobu než se předpokládalo a celý vývoj se tak značně opozdil. Především z důvodu mnoha obstrukcí ze strany výrobců.

---

[3] `http://ww1.microchip.com/downloads/en/DeviceDoc/KSZ8863MLL_FLL_RLL_DP1.8.zip` (dostupné k 20. dubnu 2018)

Po prozkoumání existujících výrobků a nabídek dodavatelů elektronických komponent padla první volba na senzor *OV4686* výrobce *OmniVision.* Jeho hlavní výhodou je zejména přítomnost čtvrtého barevného kanálu, který reprezentuje intenzitu světla ve vlnových délkách blízkého infračerveného záření NIR [22]. Touto vlastností v kombinaci s infračerveným přísvitem by se zajistilo ideální snímání ve špatně osvětlených prostorách či v noci. Bohužel, dokumentace k čipu není k dispozici bez podepsání dohody o mlčenlivosti. Z komunikace s obchodním zastoupením pro Evropu vyplynulo, že dohodu musí schválit vedení společnosti v USA a to povolení nevydá, pokud nebude garantován odběr ve vysoké kvantitě (minimálně v desítkách tisíc kusů). I přes argument, že snímač je možné volně zakoupit na internetových obchodech renomovaných prodejců komponent a měla by tedy být přístupná dokumentace. Následovala stručná odpověď - dokument by měl mít k dispozici prodejce a po vzájemné dohodě při nákupu komponenty jej vydat. Nebylo však velkým překvapením, že prodejce odmítl dokument (zřejmě kvůli vlastní dohodě o mlčenlivosti) vydat. Tímto okamžikem byla další snaha o použití jakéhokoliv snímače této společnosti ukončena.

Jedinou alternativou tak zůstaly snímače od výrobce *ON Semiconductor* (před akvizicí *Aptina Imaging Corporation*), neboť jiné v podobné cenové hladině nebyly dostupné a dražší zpravidla nemají digitální rozhraní nebo mají rozhraní paralelní. Musel by se tedy řešit převod na rozhraní *CSI.* Navíc mají velké rozměry a bylo by nutné řešit kvalitnější a rozměrnější optiku. Tím by se celý produkt značně prodražil a nebyl by ani přiměřený účelu. Po krátkém průzkumu se jevil být nejideálnější volbou obvod *AR0330*, který jsme nakonec využili. Hlavním důvodem této volby měla být kromě množství otevřených projektů na internetu zejména podpora v jádře operačního systému Linux. Ke snímači *AR0330* byla k dispozici alespoň základní dokumentace (neobsahovala kompletní mapu registrů, pouze doporučené hodnoty pro inicializaci) včetně referenčního zapojení [23]. Tak byl výběr snímače hotov.

4.5.2 Snímač AR0330

Snímač *AR0330* je optického formátu 1/3 palce. Jeho aktivní plocha činí 2304×1536 pixelů. Každý pixel je tvořen čtvercovou plochou o straně 2.2 µm. Jedná se o barevný snímač, obsahuje tedy Bayerův filtr (konkrétně v konfiguraci GRBG). Nabízí výstupní sběrnice *CSI*a *HiSPI.* Je k dispozici v dobře osaditelném pouzdře *CLCC48.* Mezi další výhody pak patří podpora externího blesku a závěrky, integrovaný fázový závěs pro generování potřebných kmitočtů, automatická korekce úrovně černé barvy a možnost synchronizace více čipů (hodí se zejména pro stereoskopické snímání). V režimu FullHD (1080p) poskytuje snímkovací frekvenci až 60 Hz. Ke svému chodu vyžaduje dvojici napájecích napětí 1,8 V a 2,8 V.

4.5.3 Zadní panel modulu

Zásadní částí modulu je zadní panel modulu, který obsahuje zdroje potřebných napětí a snímač samotný. Jedná se o přípojný bod z hlavní desky.

Modul je napájený z 12 V zdroje na hlavní desce kamery. Toto napětí je dále rozvedeno na případný čelní panel a spínací výkonový tranzistor, který může sloužit například pro výhřev optiky proti zamlžování. Také je z něj obvodem *LM22675MR-ADJ* vytvářeno napětí 3,3 V, které slouží k napájení přídavných snímačů a pomocného mikrokontroléru pro jejich obsluhu a zprostředkování dat do SOM. Z 3,3 V jsou pak dvojicí lineárních stabilizátorů vytvářena napětí 2,8 V a 1,8 V (v tomto pořadí) pro napájení samotného snímače.

Dále je ve schématu zahrnuta i komponenta pro montáž objektivu se standardním závitem C-Mount, který je určen pro čipy s rozměry 1/3 palce. Nutnou součástí je také oscilátor 24 MHz poskytující základní frekvenci pro fázový závěs a jádro snímače.

Konfigurace senzoru probíhá po sběrnici I^2C, která však využívá napěťových úrovní 3,3 V logiky. Proto je zde také umístěn převodník úrovní *PCA9306*, který pro snímač mění napěťové úrovně na 1,8 V.

4.5.4 Přední panel modulu

Přední panel modulu se k zadnímu připojuje pomocí šestnáctivodičového plochého kabelu konektory *MicroMatch*.

Na této desce je umístěn zejména infračervený přísvit složený z 30 LED diod *TSHG6200* elektronicky zapojených do matice 5×6 (na desce rozmístěných do kružnice), které dohromady poskytují optický výkon 5400 mW/sr v úhlu ±10°. Zejména kvůli nim je deska samostatná. Je možné ji posunout více dopředu tak, aby nedocházelo k zastínění přísvitu použitou optikou. LED jsou řízeny pomocí levného spínaného proudového zdroje *AL8860WT-7*, který mimo jiné umožňuje regulaci signálem s pulzně šířkovou modulací. Ten je přiveden z mikrokontroléru na zadním panelu.

Dále je zde obsažen senzor *TCS34003FNM*, který obsahuje snímače osvětlení červeného, zeleného, modrého i infračerveného světla. Může tak poskytovat prostřednictvím sběrnice I^2C do mikrokontroléru informaci o aktuální hladině osvětlení. Lze ji využít například k automatické regulaci intenzity přísvitu nebo propagovat dále do CMOS snímače či SOM pro automatickou regulaci expozice.

K desce lze připojit šestivodičovým plochým kabelem skrze konektore *MicroMatch* další pomocné senzory. K tomuto účelu je zde umístěn stejný převodník úrovní, jako na hlavní desce. Na přídavné senzorové DPS (je zahrnuta v čelním panelu) se pak nachází mikrofon *ABM-713-RC* a snímač vlhkosti (a teploty) *SHTC1* sloužící jako zdroj dodatečných informací pro případné odmlžování či vyhřívání optiky. Tato deska je oddělená z toho důvodu, aby mohla být umístěna ve venkovním prostředí a například izolační hmotou oddělena od prostředí vnitřního.

5 Konstrukce elektroniky kamery a mechanická kompletace

V okamžiku dokončení schematického zapojení je nezbytné vytvořit konkrétní podobu desek plošných spojů. Tato kapitola shrnuje volbu mechanických rozměrů a také, čemu autor DPS musí věnovat při návrhu zvýšenou pozornost. Všechny vytvořené DPS jsou součástí elektronických příloh. Při návrhu DPS bylo hojně využíváno rad a hodnot z oficiální příručky pro návrh DPS „Layout Design Guide“ [7] a proto je na ni v rámci kapitoly často odkazováno slovem „příručka“.

5.1 Mechanický návrh DPS

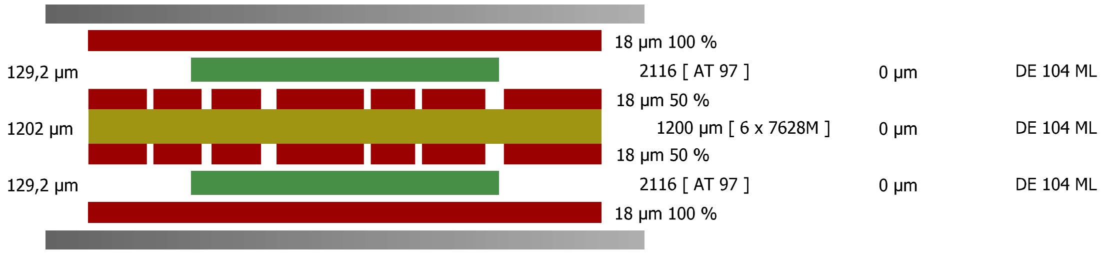
První fází je stanovení mechanických rozměrů. Jelikož účelem výrobku řešeným v této fázi je zejména demonstrace a rychlé prototypování jeho derivátů, nebyly na mechanickou stránku desky kladeny žádné konkrétní mechanické požadavky. Rozměry jsou proto relativně velké (213×90 mm) a to tak, aby se daly pohodlně protáhnout všechny spoje na deskách jen se 4 vrstvami mědi, a také bylo možné pohodlně umístit rozšiřující moduly (včetně nejdelší verze M.2 SSD disků o délce 80 mm). Bohužel zejména kvůli vyvedení signálů z jedné strany modulu nezbylo místo pro montážní otvory pro upevnění chladiče modulu z jedné jeho strany. Vhodným řešením tohoto nedostatku se stal přítlak pomocí mechanického rámu výrobku.

Pro desky modulů byly zvoleny takové rozměry, aby je bylo možné snadno zasadit do demonstračního kompletu. Proto jsou všechny „uživatelské“ konektory přístupné na zadní straně výrobku, konektor pro kamerový modul na čelní straně výrobku a konektor I/O modulu na spodní straně výrobku (předpokládá se případné umístění modulu pod hlavní deskou pravděpodobně v lyžinách).

Ethernetové moduly jsou navrženy tak, aby je bylo možné zasadit k zadní stěně výrobku do vrchní strany hlavní desky (konektory tak leží v jedné rovině). Mají rozměry 70×35 mm. Na obou panelech kamerového modulu jsou umístěny dva otvory pro vzájemné spojení a uchycení k rámu nebo čelnímu krytu pomocí šroubů a distančních sloupků.

5.2 Design a výroba DPS

Nyní přichází na řadu samotná tvorba desky plošných spojů. Na základě zkušenosti získané během tvorby této práce důrazně doporučujeme zvolit před samotným návrhem výrobce DPS, kontaktovat jej a na základě poskytnutých informací navrhnout vrstvení desky - počet vrstev, tloušťky mědi a materiál a tloušťky izolantu (tzv. prepregu). Doporučená rozložení a využití vrstev jsou v příručce dostupná v rámci kapitoly *PCB Stack-Up*. V opačném případě se lze ocitnout v situaci, kdy výrobce není schopen požadované vrstvení nabídnout (nebo pouze za příplatek či delší dobu výroby). K této situaci také došlo při zadávání výroby hlavní desky, kdy tradiční výrobci, kterých naše společnost využívá, nemají konkrétní vrstvení desky standardně v nabídce. Rozhodovalo se tedy mezi nejbližší možnou dostupnou alternativou a nebo zhotovením desek u jiného výrobce, avšak za vyšší cenu. Nakonec, po dohodě s vedoucím práce, jsme zvolili



Obrázek 6: Specifikace vrstvení DPS navržená výrobcem

drobný risk a nechali vyrobit desku v dostupné variantě. Cesty na hlavní desce jsou tak impedančně mírně nepřizpůsobené, avšak při práci s kamerou nebyl zjištěn žádný problém a vše funguje zcela korektně. V případě uvedení výrobku z fáze prototypu je však nutné tloušťky spojů poupravit pro konkrétní vrstvení (zejména také kvůli případnému elektromagnetickému vyzařování). Desky modulů vznikaly později a jsou již správně.

Použité vrstvení všech desek, tak jak bylo navrženo jejich výrobcem během specifikace zakázky, je znázorněno na obrázku 6 a případný zájemce o výrobu desky by jej měl co nejpřesněji dodržet. Červeně jsou vyobrazeny vrstvy mědi, čistě zelenou barvou vrstvy prepregu, uvnitř („špinavě“ zelenou barvou) se nachází jádro a na vnějších stranách šedě vyobrazené nepájivé masky. Celková tloušťka hotové desky je v důsledku slisování na konci výroby přibližně 1,5 mm.

Máme-li zvoleno vrstvení desky, je třeba stanovit druh vedení a rozměry spojů a mezer, aby byly co nejlépe dosaženy požadované impedance na vysokofrekvenčních spojích jednotlivých sběrnic. K dispozici je za tímto účelem i několik relativně jednoduchých on-line kalkulaček[4]. Pro úvod do problému je opět možné využít příručky, konkrétně kapitoly *Trace Impedance*. Jak již bylo zmíněno, je však vhodné nespoléhat se na uvedené hodnoty a přizpůsobit je vlastnímu vrstvení.

V tomto okamžiku je jíž možné pustit se do samotného kreslení DPS. Je však nutné klást zvýšený důraz na vzájemnou polohu a délky jednotlivých vodičů. Zejména u vysokofrekvenčních sběrnic jsou délky kritické a je třeba typicky dosáhnout shody v řádech desítek až malých jednotek mikrometrů v rámci dvou vodičů jednoho diferenciálního páru a shody typicky okolo jednoho milimetru mezi páry v rámci sběrnice. Toho je dosaženo tvorbou takzvaných meandrů. V aplikaci *Eagle* existuje pouze základní nástroj s tímto pojmenováním, který tvoří meandry v rámci jednoho diferenciálního páru tak, aby se co nejvíce přiblížil shodné délce těchto vodičů. Případně je možné zadat požadovanou délku na kterou budou oba vodiče dorovnány. Je tedy nutné odhadnout délku nejdelšího páru a na tu sběrnici přizpůsobit. Zvolili jsme postup, kdy nejprve došlo ke srovnání délky v rámci páru a následně, na předem připravených volných místech, i dorovnání délek mezi páry na jednu dostatečně shodnou. Tak se podařilo poměrně přesně dodržet náročné požadavky na jednotlivé sběrnice. Situace je o něco komplikovanější v případě nediferenciálního vedení (zde konkrétně v případě sběrnice SDIO), kde je nutné pracovat po jednotlivých vodičích. Délky cest si může uživatel v aplikaci *Eagle* zobrazit pomocí ULP skriptu s názvem *length.ulp*.

---

[4] Například zde `https://www.eeweb.com/tools/microstrip-impedance`

Požadavky na impedance sběrnic i délky párů (včetně maximální délky vodičů) jsou uvedeny v příručce individuálně pro každou sběrnici samostatně i s případnými dalšími omezeními. Ne vždy je však možné požadavky zcela dodržet (zejména kvůli místu na desce) a je nutné udělat kompromis. V tomto případě se doporučuje postupovat prioritně od nejrychlejších a nejcitlivějších sběrnic po nejpomalejší. Konkrétní pořadí (od nejprioritnějšího) doporučené příručkou je následující:

1. PCI Express,
2. USB 3.0 Super Speed,
3. SATA,
4. Ethernet,
5. HDMI a LVDS,
6. USB 2.0,
7. SDIO,
8. Paralelní rozhraní,
9. Analogové signály a audio,
10. Pomalé sběrnice (I^2C, SPI, UART, aj.).

Po dokončení spojů byl na horní stranu dodělán návrh potisku pro zvýšení přehlednosti umístění komponent. S ohledem na cenu byl zvolen jednostranný potisk a tak jsou veškeré informace viditelné ihned při pohledu z horní strany desky. Komponenty na dolní straně jsou označeny přerušovaným rámečkem.

V okamžiku doručení vyrobených desek plošných spojů bylo nutné osadit součástky. Veškeré desky jsou zejména kvůli ceně prototypu, ale i dalších organizačních příčin, osazeny ručně. Většinou se jednalo o osazování horkým vzduchem do pájecí pasty, výjimečně pájecí stanicí a cínem. U finálního výrobku by již osazování probíhalo na automatech.

Také lze pro výrobu finálního výrobku doporučit služeb výrobce DPS, který nabízí službu takzvané řízené impedance, kdy přizpůsobí desku pro konkrétní výrobní sérii tak, aby byly přesně dodrženy impedance cest.

5.3 Finální montáž prototypu

Pro prezentaci prototypu bylo nutné zhotovit jednoduchý rám, aby nedošlo k poškození výrobku (zejména flexibilních vodičů) při manipulaci. Jedná se o pět minimalistických dílů zhotovených technologií 3D tisku z materiálu PLA. Jedná se o následující části:

- Dvojice *bočních kolejnic* sloužící k uchycení hlavní desky (pravá kolejnice má při pohledu z čelní strany výstupky pro přítlak chladiče SOM).
- *Čelní rám* k uchycení modulu kamery a zajištění stability výrobku (mohlo by dojít k převažování těla kamery použitou optikou), ke kolejnicím se uchycuje dvojicí šroubů M3[5] a svazuje je dohromady z čelní strany.
- *Zadní lišta*, která se uchycuje identicky jako čelní panel a svazuje kolejnice ze zadní strany.
- *Podpora Ethernetových modulů*, zajišťuje ochranu proti vyvrácení modulů v důsledku zátěže vodiče vedoucího z konektoru modulu. Uchycuje se montážními šrouby společně s modulem k hlavní desce. Správná pozice je udána zámky v modulu.

Poslední nezbytnou komponentou je pak objektiv. Jelikož v rámci prototypu nebylo nutné dosáhnout vysoké kvality obrazu, ale spíše ověřit funkci zařízení a volbu ohniska, byl s ohledem na cenu vybrán generický objektiv od čínského výrobce[6]. Jedná se o zoom objektiv s ohniskovou vzdáleností 3.5 mm - 8 mm, regulovatelnou clonou a ostřením (vše manuální). Záměrně byl zvolen objektiv s těmito funkcemi, aby bylo možné jej používat co nejuniverzálněji. Obraz dle očekávání není z nejkvalitnějších, avšak lze však využít jakýkoliv jiný objektiv určený pro bajonety typu *C-Mount* a *CS-Mount*. Do produkčního zařízení tak bude muset být zvolena patřičná kvalitní optika odpovídající požadavkům místa nasazení.

---

[5] metrický závit o průměru 3 mm

[6] Konkrétně se jedná o následující výrobek: `https://www.aliexpress.com/item/3-5-8mm-CS-lens-CS-Mount-Varifocal-Manual-Iris-CCTV-Lens-for-CCTV-Security-Cameras/32782026024.html` (dostupný k 28. dubnu 2018)

6 Operační systém

Ačkoliv zvolený výpočetní modul obsahuje operační systém již z výroby, Není jeho použití natolik jednoznačné, jak by se mohlo na první pohled zdát. V následujících podkapitolách bude tedy rozebrána samotná volba operačního systému a také následná integrace optického snímače do jeho jádra.

6.1 Volba operačního systému

Důležitou částí projektu je volba operačního systému. Pro platformu *Tegra* je nejrozšířenější podpora operačního systému *Linux*. Ten je také dodáván přímo v rámci modulu z výroby v integrované paměti flash.

Na výběr je zde prakticky ze dvou variant. Buď bylo možné využít dodávaný systém anebo lze použít společností *NVIDIA* oficiálně modifikovanou verzi distribuce *Ubuntu* (nazvanou *Linux for Tegra*). Poslední verze s podporou procesoru *Tegra K1* je verze *R21.6* [24].

Systém integrovaný v modulu skýtá řadu výhod. Tou hlavní je optimalizace a připravenost pro platformu *Apalis*. Jedná se o variantu distribuce *Angström Linux*, která je vyvíjena řadou vývojářů otevřených hardwarových projektů (jako je například *OpenEmbedded* či *OpenZaurus*). Důležitým rysem tohoto operačního systému je oficiálně udržovaná kompatibilita s projektem *Yocto*, o němž a výhodách, které přináší, se zmíním v následující podkapitole 6.2. Dalším plus je pak minimálnost celého systému.

Naproti tomu *Linux for Tegra* je stavěný na míru originálním kitům *Jetson*. V případě jeho použití by bylo nutné nejprve se zaobírat kompatibilitou s modulem *Apalis*. Existuje sice návod[7] od výrobce modulu, jak připravit vhodný OS, ale jedná se o poměrně nepříjemné řešení. Je totiž nutné spojit obě zmíněné distribuce na úrovni archivovaných souborů kořenového souborového systému Linux. Největším problémem je zde dlouhodobá udržitelnost takového řešení. Hlavním argumentem pro použití *Linux for Tegra* je přítomnost standardního správce balíčků *dpkg* a celé vývojové platformy *CUDA*. Šetří se tak podstatně čas potřebný při přípravě k vývoji.

Po zkušenostech z předchozího projektu v rámci společnosti a následné dohodě s vedoucím práce bylo rozhodnuto o použití oficiální distribuce, a to zejména pro výhody projektu *Yocto* a také pro to, že nebylo nutné měnit operační systém celého modulu a riskovat tak jeho poškození. Tato volba je z hlediska tvorby práce diskutabilní. Na jednu stranu volba přinesla prodloužení doby vývoje (rozebráno v kapitole 8.1), na druhou stranu tato volba přinesla lepší možnosti verzování a přizpůsobení operačního systému na míru řešení, což je z hlediska udržitelnosti značné plus.

Oba systémy využívají stejné jádro (ve smyslu kernelu operačního systému), a to to oficiální z distribuce *Linux for Tegra*, které zahrnuje nezbytný kód pro využití všech nabízených možností platformy.

---

[7] `https://developer.toradex.com/knowledge-base/installing-nvidia-jetpack-with-l4t-on-apalis-tk1` (dostupný k 20. dubnu 2018)

Vzhledem ke zkušenostem z realizace zařízení by vývojářům pracujícím na projektech využívajících *CUDA* v rámci modulu *Apalis TK1* pomohlo připravit software na oficiální verzi operačního systému *Linux for Tegra* a poté jej adekvátně integrovat do prostředí distribuce *Angström* zařazením do vrstev *Yocto*. V nejideálnějším případě se oplatí mít dva samostatné kusy výrobku - jeden pro ladění a druhý pro integraci.

6.2 Yocto Project

Jak je zmíněno v hesle *Yocto Project*, nejedná se o linuxovou distribuci v pravém slova smyslu. Jde spíše o prostředek, jak si vytvořit vlastní distribuci na míru.

Projekt je otevřený a podílí se na něm velké množství subjektů včetně výrobců procesorů a procesorových modulů či koncových zařízení, což zajišťuje kontinuálně kvalitu projektu (zejména aktualitu balíčků a jejich integraci) a případné rychlé vyřešení problému.

Systém je sestavený z vrstev označovaných *meta layers*. Každá z těchto vrstev poskytuje řadu souborů nazývaných recepty - *recipes*, kde každý z nich reprezentuje nějaký konkrétní postup pro sestavení balíčku, balíku vícero balíčků nebo celé distribuce. Uživatel stanoví v konfiguraci prioritu těchto vrstev (recept z prioritnější vrstvy přepíše recept se shodným názvem z méně prioritní vrstvy) a tím docílí požadované kombinace balíčků. Na závěr pak již stačí spustit sestavení příkazem `bitbake jmeno_receptu`.

Tento přístup velmi usnadňuje kroky vedoucí k sestavení systému. Dalším přínosem je jednoduchost verzování. Společnost udržuje pouze svoje recepty (a případně některou konkrétní verzi ostatních potřebnou k sestavení systému), což přináší zejména snížení nároků na úložný prostor využívaného verzovacího systému. Taktéž je možné vytvořit a archivovat konkrétní verze balíků v podobě tarballu pro případ, že je třeba uchovat v provozuschopném stavu konkrétní verzi systému včetně možnosti integrovaný software dále modifikovat.

Nevýhodou je pak náročnější tvorba vlastních receptů, která vyžaduje dobrou znalost všech aspektů systému a základní programátorské schopnosti. Avšak lze předpokládat, že vlastní recepty budou integrovat právě vývojáři. Ti tedy musí nejprve nastudovat dokumentaci, pak je již práce ze systémem „jednoduchá“.

Více informací mohou případní zájemci získat prostudováním webu projektu, zejména pak článkem, který rozebírá, zda je pro daný účel použití vhodné [8].

Konkrétní postup pro sestavení distribuce *Angström* pro moduly společnosti *Toradex* pak lze získat ze znalostní databáze výrobce[9]. Jak je možné pozorovat, jedná se ve většině dílčích úkonů o krátké posloupnosti příkazů. Při dodržení všech pokynů je výstupem obraz operačního systému shodného s tím, jenž je zahrnut na modulu (pravděpodobně však v aktuálnější verzi).

---

[8] `https://www.yoctoproject.org/is-yocto-project-for-you/` (dostupný k 20. dubnu 2018)

[9] `https://developer.toradex.com/knowledge-base/board-support-package/openembedded-(core)` (dostupný k 20. dubnu 2018)

6.3 Integrace snímače AR0330

Zřejmě nejkritičtějším bodem pro realizaci projektu bylo zprovoznění optického snímače v rámci operačního systému. Ačkoli se během výběru snímače zdálo, že má v jádře systému Linux dobrou podporu, nebyla tato úloha tak triviální, jak se na první pohled zdálo. V důsledku si zprovoznění snímače vyžádalo v již tak časově napjatém termínu téměř tři pracovní týdny. Tato integrační fáze projektu si vyžádala více času, než bylo v plánu, a to z důvodů, které byly zapříčiněny v první řadě peripetiemi s výběrem snímače (jak bylo zmíněno v kapitole 4.5.1), tak chybným návrhem kamerového modulu. Na první verzi DPS kamerového modulu bylo zrcadlově převráceno připojení součástky v důsledku nejasného nákresu zapojení vývodů v jeho dokumentaci - jednalo se o jediný nákres z opačného pohledu a bez odpovídajícího označení.

V této podkapitole bych rád přiblížil poznatky o tom, jak je v jádře operačního systému komunikace se snímačem realizována, a dále specifika jeho integrace a použití. Výsledkem činnosti popsané v rámci této kapitoly je patch soubor jádra, který zprovozní senzor *AR0330* na platformě *Tegra K1*. Ten je přiložen k práci v elektronické podobě.

6.3.1 Počáteční stav

Při prvním připojení oživeného kamerového modulu, modifikaci „device tree“[10] a sestavení jádra s jadernými moduly snímače se však ve výpisu *dmesg* neobjevovala žádná zásadní informace o inicializaci zařízení. Experimentováním a diskuzemi na fóru výrobce se začalo rýsovat řešení problému - od moderátora fóra byly obdrženy zásadní, před tím nejspíše přehlížené informace.

Ve zdrojových kódech kernelu existují dva různe „stacky“ pro obsluhu snímačů[25]. Jeden z nich je optimalizovaný pro platformu *Tegra* a přistupuje se k němu prostřednictvím vlastního aplikačního rozhraní. Ten je však plně podporován až od generace procesorů *Tegra X*. Lze tedy použít pouze druhý z nich, který zpřístupňuje snímač druhou (a ne tak efektivní) cestou prostřednictvím standardního rozhraní knihovny *Video for Linux 2*.

Správná kombinace modulů je tedy *tegra_camera* a *ar0330_v4l2*. Principiálně je celý systém sestaven tak, že jako první se zavedou jaderné moduly snímačů, které provedou jejich inicializaci sběrnicí I^2C (např. právě *ar0330_v4l2*). Následně se zavede jaderný modul *tegra_camera*, který vyhledá inicializované moduly snímačů a vytvoří pro ně rozhraní v uživatelském prostoru systému.

6.3.2 Zavedení modulu

Po zavedení korektních modulů se ve výpisu jádra začala objevovat chyba poukazující na problémy s komunikací po sběrnici I^2C.

Ve výpisech byla uvedena špatná adresa snímače na sběrnici a nedařilo se ji však změnit žádnou úpravou souboru „device tree“. Řešení se ukázalo být přímo v jádře v místě, kde

---

[10] Soubor sloužící k popisu periferií a jejich vztahů v jádře systému, jehož úkolem je zejména správná inicializace hardwarových komponent a jejich zpřístupnění v OS.

dochází k poskytnutí informací o připojených snímačích. Jedná se o soubor „arch/arm/mach-tegra/board-apalis-tk1-sensors.c“. To tedy znamená, že pro jakýkoliv další připojený senzor je nezbytné překompilovat celé jádro. Snímače zmíněné v „device tree“ slouží pouze pro nové (tedy na této platformě nepodporované) rozhraní. Po drobných modifikacích senzor začal s modulem komunikovat alespoň po konfigurační sběrnici.

Ne vždy však byla konfigurace úspěšná, nyní bylo tedy zapotřebí pátrat po problémech na úrovni signálů sběrnice. Jelikož se jednalo o sběrnici, která byla výrobcem modulu určena jako dedikovaná pro tento účel, dalo se předpokládat, že na sběrnici proběhne nějaký pokus o komunikaci. Tvrzení se potvrdilo po připojení na logický analyzátor, avšak spolehlivost byla nízká. Následovaly pokusy o opravu elektroniky a spíše náhodou byla objevena závada způsobená flexibilním kabelem - pravděpodobně docházelo v rámci rozhraní kabelu a konektoru k propojení vodičů sběrnice s napájecími vodiči. Tím pochopitelně byla znemožněna činnost sběrnice, naštěstí však nebyl nijak poškozen modul samotný (zřejmě díky ochranným rezistorovým polím na těchto signálech u konektoru modulu). Řešením se ukázala být výměna kabelu, jež byl nějakým nepostřehnutelným způsobem poškozen. Nicméně i tak si odhalení této chyby vyžádalo nemálo času.

6.3.3 Sběrnice MIPI a konfigurační registry senzoru

Další rozsáhlou kapitolou práce se senzorem bylo zprovoznění získávání dat ze snímače.

V tomto okamžiku začalo být zařízení dostupné v operačním systému jako „/dev/video0“. Po spuštění jednoduchého streamování docházelo okamžitě ke zhroucení celého systému doprovázeného chybou „syncpt timeout“ a výpisem registrů periferie obsluhující sběrnici *MIPI CSI*, což poukazovalo na problémy s datovou komunikací po sběrnici. Tento názor potvrdil i pohled do zdrojových kódů a diskuze na fóru společnosti *NVIDIA*. Zde bylo nalezeno doporučení (dokonce pro identický snímač), které doporučovalo v inicializaci modulu *tegra_camera* (konkrétně v souboru „drivers/media/platform/soc_camera/tegra_camera/vi2.c“) upravit hodnotu jednoho z registrů procesoru následujícím makrem:

```
TC_VI_REG_WT(cam, TEGRA_VI_CFG_VI_INCR_SYNCPT_CNTRL, (1 << 8));
```

Tím nebylo docíleno řešení problému (které však nebylo zodpovězeno ani v onom vlákně diskuzního fóra), ale tento konfigurační bit zajistil, že v případě vypršení vyhrazeného času pro čtení snímku nedojde k zablokování jádra, ale pouze k vyhlášení chyby [26].

Nevyhnutelné bylo rozhodnutí zkontrolovat signály sběrnice. Jelikož se jedná o sběrnici vysokorychlostní (hodinový kmitočet v řádech stovek MHz v závislosti na rozlišení obrazu a počtu použitých datových párů), bylo velmi obtížné provádět s dostupným vybavením na takové sběrnici měření. Navíc se jedná o paketovou komunikaci. Pro hlubší analýzu je tedy zapotřebí vlastnit velmi rychlý logický analyzátor a dekodér komunikace. Takové vybavení však nebylo k dispozici. Muselo tedy dostačovat velmi přibližné a zkreslené měření za využití osciloskopu, kterým proběhla kontrola napěťových úrovní a průběhu signálů. Signály se zdály být v pořádku a nějaká data proudila.

Zajímavým poznatkem bylo, že se ve výpisu jádra objevovaly stále stejné stavové kódy v registrech periferie. Výsledkem hlubšího pátrání bylo nalezení informace o nezbytnosti nastavení přesného časování sběrnice na obou jejích koncích a následovala další trpělivá práce za účelem nalezení funkční konfigurace registrů kamery v jiných otevřených projektech včetně těch, které nevyužívaly procesor *Tegra*. Byl tak objeven projekt *physical-hasher* [27], který v rámci dílčího projektu *bitmark-microscope* využívá stejný senzor a dokonce zde byla nalezena i starší verze oficiální dokumentace senzorů s mapou registrů. Díky tomu se po několika pokusech podařilo docílit korektního nastavení a video stream začal pracovat.

6.3.4 Finální úpravy modulů

V poslední fázi proběhly již jen drobné úpravy jaderných modulů. Zejména se jednalo o doplnění nepodporovaných formátů dat s barevným filtrem (CFA, zde konkrétně *Bayerův* filtr GRBG).

Dále pak bylo nezbytné dotvořit mazání chybových příznaků ve stavových registrech periferie, jinak docházelo při vyšším zatížení procesoru ke ztrátě snímku, ze kterého se však procesor již nezotavil a zobrazoval stále dál stejnou chybou, jako v předchozí kapitole. Stream v tomto případě pokračoval přibližně pěti snímky za vteřinu, což byla doba udaná limitním časem čekání na synchronizační bod. Tento problém bylo zpravidla možné vyřešit odebráním a opětovným zavedením jaderného modulu *tegra_camera* pomocí nástroje *modprobe*.

7 Příprava analytické funkce kamery

7.1 Volba funkce

Od samotného začátku projektu bylo záměrem, že funkcí pro ověření výkonu kamery bude rozeznávání osob v obraze, nejpravděpodobněji s využitím neuronových sítí. K tomuto účelu však musí být zvolena a následně vytrénována vhodná neuronová síť.

7.2 Umělé neuronové sítě

Podstatná část práce závisí na strojovém učení za pomocí neuronových sítí. Je tedy záhodno přiblížit, o co se ve skutečnosti jedná.

Umělá neuronová síť je inspirována neurony a jejich propojením v živých organismech. Je složena z množství neuronů, do kterých vstupují hodnoty. Tyto hodnoty jsou zpracovány aktivační funkcí neuronu, a ta v podobě jedné číselné hodnoty poskytuje výsledek dále. V nejjednodušším případě jsou tyto umělé neurony organizovány do vrstev, kde jednotlivé neurony vrstvy mají na vstupech výsledky všech neuronů vrstvy předchozí a výsledek poskytují všem neuronům vrstvy následující. V rámci jedné vrstvy neurony nikdy nejsou propojeny. Takový typ vrstvy se označuje jako *Fully connected layer*. Výjimkou jsou vstupní a výstupní vrstva, kde jsou hodnoty dodávány uživatelem (nebo jsou mu poskytovány).

Pro správnou funkci je nutné neuronovou sít takzvaně natrénovat pro konkrétní účel. Jedná se de facto o tvorbu statistického modelu na základě poskytnutých dat. V případě plně propojených sítí se upravují váhy přenosových funkcí jednotlivých neuronů pomocí technik numerické optimalizace, nejčastěji využitím metody *Stochastic gradient descent*.

V případě obrazových dat lze však velmi často narazit na sítě nazývané jako *konvoluční*. Jak již název napovídá, taková síť je složena z vrstev, kde jsou neurony propojeny tak, aby reprezentovaly matematickou operaci konvoluce. Každá vrstva je reprezentována množinou neuronů reprezentující jedno či více konvolučních jader. Výstupem konvoluční vrstvy je pak, pro každé konvoluční jádro samostatně, výsledek aplikace operace konvoluce na vstupní data vrstvy. Během tréninku se pak modifikují konvoluční jádra. Lze však z definice vypozorovat, že tak dochází k neúměrnému navyšování množství dat mezi vrstvami a tím i snižování výkonu. Tomuto problému se většinou předchází záměrnou decimací dat. Ta je reprezentována speciálním druhem vrstev označovaných slovem *pooling*. Jedná se o vrstvy během tréninku zcela neměnné a provádějí zpravidla agregaci prostorově blízkých hodnot, nejčastěji zvolením maximální či minimální hodnoty nebo jejich průměrováním. Jejich vedlejším účinkem pak je poskytnutí částečné ochrany proti přetrénování[11] na konkrétní množinu dat.

---

[11] Při přetrénování síť podává velmi dobré výsledky na trénovací množině, ale není schopná korektně fungovat na datech, které v ni zahrnuty nebyly.

7.3 Klasifikace, lokalizace a detekce

Pro uvedení do problému je nutné zmínit, čím je rozeznávání osob z pohledu analýzy obrazu. Zmíněná funkcionalita se skládá ze dvou základních úloh a to *klasifikace* a *lokalizace*.

Cílem *klasifikace* je na základě charakteristických rysů v datech (zde obrazových) rozhodnout, o jaký objekt se jedná, respektive do jaké skupiny (třídy - proto název klasifikace) jej zařadit. Funkce nebo algoritmus, která rozhodování provádí se nazývá *klasifikátor*. V případě strojového učení s využitím neuronových sítí jsou vstupní proměnnou funkce typicky obrazová data. Výstupem pak zpravidla bývá vektor, kde každá jeho složka přísluší jedné třídě objektů a její hodnota reprezentuje pravděpodobnost zařazení do příslušné třídy. Laicky řečeno, vstupem do klasifikátoru může být například fotografie muže, výstupem pak v ideálním případě bude vektor s nejvyšší hodnotou na pozici náležící mužům.

Nabízí se ovšem otázka, co se stane v případě, kdy se na fotografii nachází manželský pár. V takovém případě by zřejmě očekávaným chováním byl výstup klasifikující muže i ženu se shodnou pravděpodobností. V některých případech může být tento výstup žádoucí. Chceme-li však například sledovat samostatně více osob v obraze, je nutné znát také jejich polohu.

Tímto případem se zabývá úloha *lokalizace*. Ta vychází ze znalosti třídy objektu a snaží se o jeho dohledání v obrazových datech.

Kombinací obou těchto úloh je pak *detekce*. Ta zjišťuje, co za objekty zájmu se v obraze nachází a současně se snaží rozeznat umístění jednotlivých objektů. Funkce nebo algoritmus provádějící detekci se označuje slovem *detektor*. Vstupem jsou tedy zpravidla celá obrazová data, výstupem pak například obdélníky vyznačující oblasti, do kterých jednotlivé objekty zájmu spadají a třída, do níž náleží.

7.4 Průzkum existujících řešení

Pro demonstrační účely i ověření analytických možností by bylo vhodné, aby zařízení umělo řešit problém *detekce*.

Tradiční metody řešení využívají typicky některou z variant metody *sliding window*, kdy po obraze „pohybujeme“ virtuálními okénky různých velikosti, na jejichž obsah je aplikován klasifikátor. Tato metoda je již na první pohled velmi výpočetně náročná, neboť pro kvalitní výsledek je nezbytné celý obraz prozkoumat několikrát s různými velikostmi oken. Jednotlivé oblasti se navíc vzájemně překrývají, aby se snížila pravděpodobnost selhání vynecháním nebo rozdělením objektu.

Ideální by však bylo, kdyby byl tento problém řešitelný v jednom průchodu, čímž by se značně snížily výpočetní nároky (i když nejspíše za cenu nižší spolehlivosti). Rešerše přinesla tyto výsledky týkající se zejména aktuálního stavu v této oblasti se zaměřením na neuronové sítě.

Prvním nálezem byla rodina neuronových sítí *R-CNN*, konkrétně varianta *Faster R-CNN*. Zjednodušeně se jedná o konvoluční síť, která zavádí variantu sítě *Region Proposal Networks*, která predikuje potenciální oblasti a omezuje tak množinu k prohledávání [3]. Bohužel již při čtení anotace použití této sítě nepřicházelo v úvahu, neboť byla zmiňována rychlost detekce 5 snímků za sekundu na desktopovém GPU. Navíc je síť poměrně složitá, co se týče její vnitřní struktury.

Nejvíce zmínek v mnoha srovnáních však bylo o sítích *SSD* (Single shot detector) a dále síť *YOLO* (You only look once), která byla nakonec zvolena. Na oficiálním webu autora této sítě je možno nalézt graf a tabulku srovnávající výkon a přesnost obou těchto sítí s mnoha dalšími [1]. Avšak jako nejzajímavější se jevila v tabulce zahrnutá varianta sítě nazvaná *Tiny YOLO* a to zejména kvůli nízkému počtu výpočetních operací. Další práce tedy vycházela z této alternativy.

7.5 Neuronové sítě YOLO

Z názvu sítě *You only look once* je zřejmé, že primární myšlenkou sítě je provést celou detekci během jednoho pohledu na obrazová data, tedy reprezentovat celý výpočet jen jedním průchodem neuronovou sítí [2].

Jedná se čistě o konvoluční neuronovou síť. Jsou tedy využity pouze vrstvy *konvoluční* a *max pooling*. Detekce je v síti zajištěna speciální reprezentací hodnot ve výstupních datech. V originální síti *Tiny YOLO*[12] obsahuje poslední konvoluční vrstva 125 jader a je aplikována nad mřížkou hodnot o rozměrech 13×13. Zatímco výstupní rozměry jsou dány postupnou decimací mezi vrstvami sítě, počet jader není zvolen náhodně. Autoři sítě došli k závěru, že ideálního poměru mezi rychlostí a kvalitou je dosaženo, detekuje-li každá buňka výsledné mřížky až pět regionů. Síť je schopna klasifikovat objekty dvaceti tříd, což je reprezentováno vektorem dvaceti hodnot pravděpodobnosti. K tomuto vektoru je pak dále přibalen výsledek lokalizace reprezentovaný dvojrozměrnými souřadnicemi středu, šířkou a výškou. Poslední hodnota pak reprezentuje pravděpodobnost toho, že se na dané pozici vyskytuje vůbec nějaký objekt zájmu. Prostým sečtením a následným vynásobením těchto hodnot se dostáváme k počtu konvolučních jader poslední vrstvy.

7.6 Framework Darknet

Dalším bodem po výběru sítě bylo zvolit vhodný framework pro práci s neuronovými sítěmi. Po prozkoumání několika alternativ se jako nejvhodnější ukázalo být použití frameworku *Darknet*, který je vyvíjen přímo jedním z autorů sítí *YOLO* a vše potřebné je v něm tak obsaženo. Zásadní výhodou tohoto frameworku je jeho implementace v jazyce *C* a integrace technologie *CUDA*. Tento fakt byl ideálním do budoucna, neboť jej pak bylo možné snadno použít i na samotném procesoru *Tegra* v rámci demonstrační streamovací aplikace.

Ačkoli se jedná o spustitelný program, označení *framework* není samoúčelné. Projekt je rozdělen na zdrojové kódy v adresáři *src*, které reprezentují samostatně použitelnou knihovnu

---

[12] Ve verzi trénovanou nad datasetem VOC. Ve zbytku práce vycházím z varianty používající dataset Microsoft COCO.

a na spustitelnou aplikaci v adresáři *examples*. Kód aplikace je pak dle všeho udržovaný jen podle aktuální potřeby a není příliš čistý ani přehledný. Množství věcí je tak před použitím nutné upravit přímo v kódu.

Drobnou nevýhodou tohoto frameworku je absence oficiální verze pro operační systém Windows. Jelikož na stroji s výkonnou grafickou kartou byl k dispozici pouze zmíněný operační systém, musela se pro něj aplikace upravit spojením zdrojových kódů starší verze komunitou upravené pro Windows[13] a v té době aktuální oficiální „produkční“ verze frameworku [14]. Bylo potřeba provést ještě pár drobných úprav a podařilo se tak odhalit i malé množství drobných chyb.

Výsledný kód použitý pro vytrénování na operačním systému Windows včetně binární verze je pro reprodukovatelnost výsledků přiložen v elektronické podobě k této práci. Ve smyslu zadání jej lze chápat jako onu *podpůrnou utilitu*.

7.7 Modifikace a trénování sítě

Nejprve bylo třeba ověřit, jak se bude síť chovat na obraze poskytovaném kamerou. Na počátku trénování nebyla kamera ještě zcela provozuschopná, proto bylo použito několik namátkou vybraných obrázků ve formátu FullHD z internetu převedených do odstínů šedi. Snímač je sice barevný, ale v případě použití ve špatných světelných podmínkách s infračerveným přísvitem obraz znatelně ztrácí na barevnosti. Pro univerzálnost je tedy vhodnější pracovat s nebarevným obrazem. Při spuštění aplikace dle pokynů uvedených na oficiální internetové stránce síť poskytovala velmi nepřesvědčivé výsledky. Bylo tedy třeba vyzkoušet síť vytrénovat znova a mírně pozměnit její strukturu.

Originální síť zpracovávala čtvercový obraz 416×416 pixelů. Zejména kvůli formátu snímače bylo vhodné, aby síť měla na vstupu obraz stejného poměru stran a nedocházelo tak k deformaci objektů. Další očekávanou výhodou pak bylo zjednodušení výpočtů škálování obrazu v pipeline výrobku. Množství tříd (celkem 80), které originální síť poskytovala, bylo pro účely práce vyhodnoceno jako předimenzované. Přijatelná se zdála být redukce na 16 tříd, které by mohly být v dohledovém systému zajímavé. Konkrétně se jednalo o třídy „person“, „bicycle“, „car“, „motorbike“, „bus“, „truck“, „dog“, „horse“, „backpack“, „umbrella“, „handbag“, „suitcase“, „bottle“, „knife“, „laptop“ a „cell phone“. Poslední úpravou byla zmiňovaná redukce z tří barevných kanálů na jeden.

Tiny YOLO bylo vytrénováno na datasetu Microsoft COCO (Common Objects in Context) [4]. Jedná se o datovou sadu obsahující přes 82700 trénovacích obrázků a 40500 validačních. Jak je patrné z názvu, obsahuje zejména snímky běžných objektů zachycených v množství rozličných kontextů. Nejčastěji zastoupeným objektem v datové sadě jsou právě lidské osoby, což je pro potřeby realizovaného zařízení ideální. Bylo však nutné vytvořit modifikovanou verzi. Za tímto účelem tedy vznikla dvojice jednoúčelových utilit v jazyce C# (jsou taktéž přiloženy v elektro-

---

[13] `https://github.com/unsky/yolo-for-windows-v2` (dostupné k 20. dubnu 2018)

[14] `https://github.com/pjreddie/darknet` (dostupné k 20. dubnu 2018)

(a) Originální snímek (b) Přizpůsobený snímek

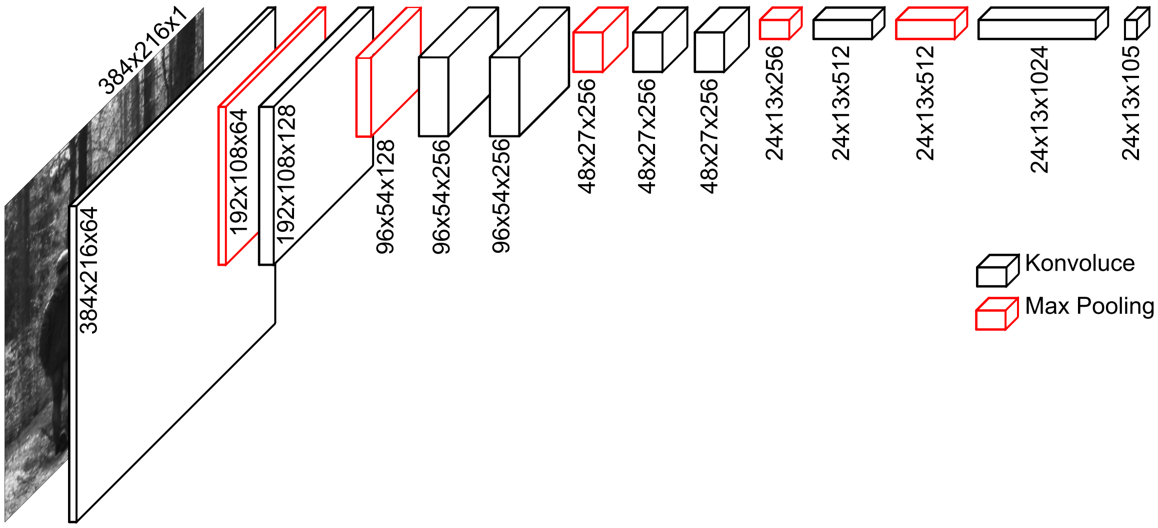
Obrázek 7: Ukázka datasetu Microsoft COCO

nické podobě). První z nich byl vstupní dataset nejprve přefiltrován jen na obrázky obsahující požadované třídy, čímž se zmenšila velikost trénovací množiny přibližně na 57000 a validační na 28000. Druhým nástrojem se poté vykonala úprava na verze s poměrem stran 16:9 (aby odpovídal snímači) a to prostým zopakováním a ořezem do prázdných míst. Stejná transformace pochopitelně proběhla současně i na popisných souborech, tak aby byly všechny zkopírované instance zahrnuty do trénování. Ukázka snímku z datové sady COCO před a po zpracování je ke shlédnutí na obrázku 7 (převzato taktéž z [4]). Každému takovému snímku pak v datové sadě přísluší popisný soubor například v následujícím formátu:

```
0 0.170624 0.503125 0.221636 0.790625
0 0.307388 0.136719 0.034301 0.273438
0 0.265172 0.135938 0.041337 0.271875
```

První sloupec označuje třídu objektu, následují středy rámečku objektu v horizontální a vertikální ose. Poslední dva sloupce určují jeho šířku a výšku.

Následně byl zahájen trénink s počátečním využitím vah z původní sítě. To však nepřinášelo očekávané výsledky a síť se příliš nezlepšovala ani po několika dnech tréninku. Sáhlo se tedy po variantě modifikovat i vnitřní strukturu sítě a začít od nuly. Zde byl již trénink úspěšný a ponechán volně doběhnout až do počtem iterací nastaveného konce.



Obrázek 8: Architektura použité neuronové sítě

7.7.1 Použitá verze sítě

Finální použitá verze sítě má rozšířen počet vnitřních vrstev po vzoru často používané klasifikační sítě *VGG-16*, tedy zopakováním stejně velkých konvolučních vrstev ihned za sebou. Tím ale velmi rychle narostly nároky na paměť i množství operací[15]. Nejjednodušším řešením bylo snížit rozměry vstupního obrazu. Jako rozumný kompromis se zdála být pětina původní velikosti 1080p. I tak bylo ale nutné částečně zredukovat i počet navržených vrstev.

Síť tedy provádí detekci nad vstupním obrazem o rozlišení 384×216 pixelů. Vnitřně obsahuje 9 konvolučních vrstev a 5 „maxpooling“ vrstev. Výstupem je mřížka o rozměrech 24×13 buněk. Každá z buněk provádí 5 detekcí a klasifikuje detekce do 16 výše uvedených tříd. Výpočet vyžaduje přibližně 19,13 miliard operací s čísly s plovoucí desetinnou čárkou. Architektura sítě je znázorněna na obrázku 8.

Trénování finální sítě zabralo přibližně 10 dní nepřerušovaného čistého strojového času na dvojici grafických karet *GeForce GTX 1060 6GB* a *GeForce GTX 750 Ti 2GB*, veškeré dílčí experimenty s modifikacemi v součtu pak přibližně dvojnásobek. Kvalita sítě byla během tréninku pravidelně kontrolována pomocí aplikace frameworku darknet následujícím příkazem:

```
darknet detector recall [konfigurace datasetu] [konfigurace site] [soubor vah]
```

Výsledkem je dvojice procentuálních hodnot *IOU* (intersection over union) a *Recall*, kterou jsem využil jako hlavní metriku pro zhodnocení kvality sítě. První z dvojice, *IOU*, udává procentuální poměr průniku plochy předpovězené oblasti ku ploše jejich sjednocení. Čím přesněji je oblast předpovězena, tím je tato hodnota vyšší. Druhá hodnota pak vyjadřuje kolik procent detekcí z celkového počtu bylo správně klasifikováno. Síť, která je předmětem této práce, dosahuje u obou hodnot na redukovaném validačním datasetu *45%*. Původní síť *Tiny YOLO* dosahovala

---

[15] Množství operací lze zjistit spuštěním aplikace frameworku darknet s argumentem *ops*

37% IOU a *35% Recall* na originálním COCO datasetu. Navržená síť tedy bude v zařízení poskytovat téměř o *10%* přesnější klasifikaci.

Procentuální úspěšnost sítě sice není nejpřesvědčivější, ale vzhledem k principu a rozměrům sítě se jedná o významný výsledek. Je však zřejmé, že se jedná o síť vhodnou spíše pro demonstraci zařízení než k produkčnímu užití. V takovém případě bude nutné síť modifikovat či zvolit jiný přístup na míru řešenému problému, včetně volby vhodné datové sady nebo sběru vlastních dat.

8 Aplikace streamování a analýzy videa

Zásadní komponentou celé práce je aplikace pracující na platformě *Tegra*, která provádí zpracování obrazových dat z kamery, analyzuje je a poskytuje dále uživateli. Bez ní by nebylo možné zhodnotit výsledek práce. Tato aplikace by měla být pokud možno co nejvíce rozšiřitelná, aby ji bylo možné při dalším pokračování projektu vhodně přizpůsobit požadavkům. V rámci řešení této práce však bylo cílem zejména ověření výkonu procesoru. Řešení se tak zaměřuje zejména na její jednoduchost, srozumitelnost a účelnost.

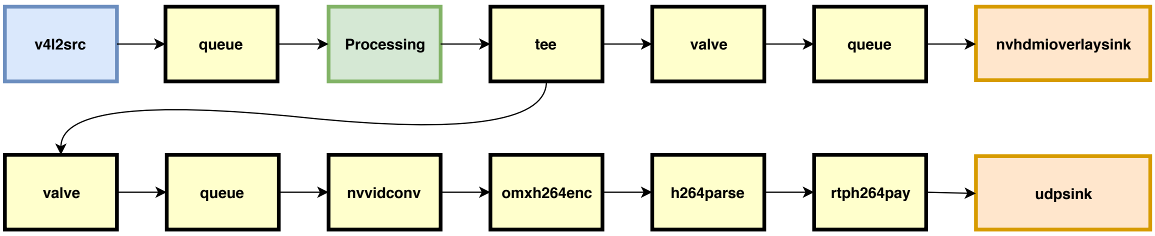
8.1 Zahájení vývoje

Ačkoli se může zdát, že se zahájením vývoje by neměl být žádný problém, byla mírně řečeno překvapením náročnost úlohy. Zásadním problémem bylo zprovoznění křížového překladu s podporou technologie *CUDA*. Procesor *Tegra K1* je 32-bitový, avšak společnost *NVIDIA* ukončila vývoj 32-bitové verze technologie *CUDA* s verzí *7.0*. Poslední podporovanou verzí pro *K1* je tedy verze *6.5* z října roku 2014. Právě tato „zastaralost“ sebou přináší jistá úskalí. Při vývoji aplikace musí programátor použít překladače *nvcc*, který ovšem interně používá standardní *GCC* v tomto případě verze nejvýše *4.8*. Ta není v současných systémech zpravidla zahrnuta a není již ani součástí vrstev v projektu *Yocto*. Zejména s ohledem na blížící se termín projektu, došlo na použití virtuálního stroje v prostředí *VirtualBox* s poslední oficiálně podporovanou verzí operačního systému Ubuntu verze *14.04*, kde bylo možné vše zprovoznit oficiálním instalátorem a z repozitářů. Do budoucna však bude nezbytné zaintegrovat celý proces do vlastní vrstvy v rámci projektu *Yocto*, aby bylo možné celé softwarové vybavení kamery dlouhodobě uchovávat v snadno udržovatelném stavu.

8.2 GStreamer

Z pohledu dalšího vývoje i stráveného času by nebylo příliš výhodné realizovat celé zpracování „na zelené louce“, byť s použitím knihoven pro jednotlivé dílčí úkoly. Na základě zkušeností získaných během realizace jednoho z nedávných projektů v rámci firmy, jsme rozhodli o použití frameworku *GStreamer*. Jedná se o otevřený a multiplatformní balík aplikací a knihoven, zaměřených na kompletní práci s multimédii. Jeho vývoj započal již v roce 2001 a nadále aktivně pokračuje až do dnešních dní (19. března 2018 byla uvolněna nová verze 1.14). Projekt je vyvíjen v jazyce C nad knihovnou *GLib 2.0*.

Princip využití spočívá v tvorbě orientovaného grafu elementů (například filtrů) nazývaný *pipeline*, které se společně propojují pomocí bodů nazvaných *pad*. Existují dvojího druhu, a to výstupní *source pad*, který data poskytuje z elementu dále, a vstupní *sink pad*, kterým do elementu data přicházejí. Každému *padu* je dále přiřazena hodnota určující, jaký typ dat je schopný generovat nebo akceptovat. Nelze tak propojit vzájemně nesouvisející elementy.



Obrázek 9: Struktura hlavní video pipeline

Silnou stránkou tohoto frameworku je jeho modularita, která umožňuje snadnou rozšiřitelnost. Nejen díky tomu je na embedded zařízeních hojně podporován přímo výrobci procesorů i komponent. Vlastní moduly poskytuje i společnost *NVIDIA* v rámci operačního systému *Linux for Tegra*. Některé z nich jsou použity i v rámci realizované aplikace. Bohužel modul umožňující zpracování dat s využitím technologie *CUDA* prostřednictvím uživatelské sdílené knihovny je dostupný až pro procesory řady *Tegra X1* a výše. Nezbylo tedy než tuto část realizovat samostatně.

8.3 Konstrukce aplikace

Aplikace je logicky členěna do několika celků. Základem je hlavní funkce aplikace obsažená v souboru *main.cpp*. Zde se nachází inicializace a obsluha pipeline, zpracování signálu pro aktualizaci konfigurace a inicializace *CUDA*. Podstatná je však zejména pipeline realizující hlavní zprostředkování obrazu ze snímače na výstupy zařízení. Pro názornost je její struktura vizualizována na obrázku 9.

Na vstupu pipeline se nachází element *v4l2src*, který získává obraz ze snímače v podobě surových mozaikovaných dat ve formátu *Bayer*. Z něj data proudí dále do elementu fronty *queue*, jež slouží primárně jako vyrovnávací buffer před vstupem do samostatného zpracování. Je-li zpracování pomalé, zařídí fronta zahazování starých snímků. Nedochází tak k navyšování alokované paměti a zpožďování obrazu o více než 100 ms, což je limitní hodnota nastavená ve zdrojovém kódu. Následuje *application-cuda-process* detailněji rozebraný v následující podkapitole 8.4. Na jeho výstupu je již barevný obraz převedný do standardního formátu I420[16]. Ten je vyžadován enkodérem a výstupním modulem HDMI. Následně je využito elementu *tee*, kterým jsou data duplikována do dvou samostatných větví.

První jednodušší větev zprostředkovává zobrazení videa na rozhraní HDMI a obsahuje elementy *valve*, *queue* a *nvhdmioverlaysink* (v tomto pořadí). Hlavní funkcí *valve* je nabídnout prostřednictvím konfigurace uživateli možnost zahazováním snímků celou větev pozastavit. Účel fronty je zde totožný. Poslední zmíněný element pak obrazem překrývá výstupní buffer zařízení HDMI.

---

[16] Reprezentace obrazových dat pomocí jedné jasové složky v plném rozlišení následované dvěma barevnými složkami o polovičním rozlišení.

Druhou větví je zajištěno odesílání videa protokolem RTP do sítě Ethernet. Počáteční část je svou funkcí i propojením zcela totožná s větví HDMI. Následuje element *nvvidconv* přesouvající data do vhodné paměti, odkud si je následně přebírá element *omxh264enc* provádějící enkódování do standardního video formátu *H.264*. Ten je následně rozparsován pomocí elementu *h264parse*, dále elementem *rtph264pay* obalen do multimediálního přenosového protokolu RTP a na závěr elementem *udpsink* dopraven prostřednictvím protokolu UDP na místo určení. V případě problémů s licencováním kodeku v produkčním výrobku je možné přejít na libovolnou podporovanou alternativu (např. VP8) jen výměnou tří příslušných elementů.

8.4 Modul zpracování obrazu

Samotné zpracování a analýza obrazu je do pipeline integrována prostřednictvím takzvaného statického zásuvného modulu nazvaného *application-cuda-process()*. Slovo „statický“ zde poukazuje na fakt, že modul není samostatnou sdílenou knihovnou, ale je součástí aplikace. Ve zdrojových kódech je reprezentován soubory *processing.cpp* a *processing.hpp*. Jeho hlavní funkcionalita je rozdělena do několika procedur. První z nich nazvaná *processing_filter_init_data()* inicializuje neuronovou síť a alokuje příslušnou grafickou paměť potřebnou pro zpracování toku obrazových dat. Opakem je pak procedura *processing_filter_free_data()*.

Nejdůležitější z nich je procedura *processing_filter_process()*. Zde je realizováno celé hlavní zpracování. Jednotlivé úkony jsou rozebrány v následujících sekcích textu. Na začátku procedury je pak navíc ověřen průběh analýzy a na jejím konci zahájena analýza nová. Jestliže byla úspěšně dokončena, jsou výsledky detekce prezentovány uživateli prostřednictvím „orámování“ lokalit barevnými obdélníky přímo v obraze (funkce *processing_filter_draw_detections()*). Do budoucna by bylo vhodné zasílat data i nějakým způsobem samostatně ve strojově zpracovatelné podobě. K tomu je však nutné stanovit požadavky na formát pro potřeby konkrétní aplikace a nemělo tedy smysl se tímto nyní hlouběji zabývat. K vykreslování rámečků je použita vektorová grafická knihovna *cairo*. Veškerý kód pro tuto funkcionalitu obsahují soubory *overlay.cpp* a *overlay.hpp*.

8.4.1 Demozaikování

Prvním krokem zpracování obrazu je takzvané demozaikování. Jeho účelem je převést surová data z obrazového snímače do některé z běžných reprezentací obrazových dat. Před světlocitlivými buňkami snímače se nachází pole barevných filtrů (zde v konfiguraci Bayer - GRBG). Každá oblast o rozměrech 2×2 obrazových bodů je tak reprezentována dvojicí hodnot pro barvu zelenou (což odpovídá citlivosti lidského oka) a po jedné hodnotě pro každou z dvojice červené a modré barvy. Je zřejmé, že pro kompletní pokrytí obrazu barevnými hodnotami je nezbytné chybějící hodnoty dopočítat lineární interpolací. Ta je reprezentována výpočetním jádrem CUDA, které je k nalezení v souboru *debayer.cu*, konkrétně pak funkci *debayer2x2Kernel()*. Jelikož pro interpolaci potřebujeme znát hodnoty navzájem blízké v obrazové reprezentaci, ale neležící ve své blízkosti uvnitř paměti, bylo zvoleno poskytovat vstupní data prostřednictvím texturovací paměti, která je pro tento typ přístupu optimalizovaná. Výstupem jádra je snímek obsahující pro každý obrazový bod všechny barevné složky.

Vedlejší funkcí demozaikovacího jádra je průměrování hodnot barevných složek zpracovávané dvojice řádků vstupního obrazu. Ty jsou následně pomocným jádrem *MeanKernel()* zprůměrovány do jedné hodnoty. Tato informace slouží jako základní vstup algoritmu vyvážení bílé. Hlavní motivací pro zahrnutí tohoto výpočtu k demozaikování byla zejména minimalizace paměťových přenosů - data jsou již načtena ve sdílené paměti pro potřeby interpolace.

8.4.2 Vyvážení bílé

Jako součást zpracování byl zahrnut i algoritmus automatického vyvážení bílé. Nejedná se sice o esenciální součást řešení, ale znatelně zlepšuje kvalitu výsledného obrazu jak pro uživatele, tak zejména pro potřeby detekce. Vzhledem k faktu, že v rámci práce nebylo řešeno automatické přizpůsobování expozice, které bude vyžadovat řadu dalších experimentů, supluje vyvážení bílé částečně také korekci nízké expozice snímku (avšak za cenu zvýšení šumu v důsledku zesílení). Pro vyvážení bílé byl adaptován algoritmus navržený na vývojářském blogu společnosti NVIDIA [28].

Vstupem do algoritmu jsou tři korekční faktory (pro každou barevnou složku samostatně). Ty byly vypočteny z průměrných hodnot barev na konci průměrovacího jádra tak, že je střední hodnota všech barevných kanálů podělena střední hodnotou každého jednotlivého kanálu samostatně. Jádro *whiteBalanceKernel()* pak jen násobením faktory aplikuje na každý obrazový bod. Implementaci lze nalézt v souboru *whitebalance.cu.*

8.4.3 Downsampling

V případě, že je aktivována i analytická funkce obrazu, přichází na řadu i jádro *subsampleKernel()* ze souboru *downsample.cu.* Neuronová síť vyžaduje na vstupu miniaturu (obraz pětinové velikosti) v šedé škále barev. Tento požadavek jádro splňuje tak, že každý obrazový bod na pozicích odpovídajících zadanému celočíselnému koeficientu zmenšení vypočte odpovídající jasovou složku podle vzorce $(0,299; 0,587; 0,114) \cdot (R; G; B)$. Takovýto obraz pak uloží do samostatného bloku paměti, který se nezmění až do následující iterace detekčního algoritmu.

8.4.4 Překryv a konverze barevného prostoru

Posledním krok zpracování obrazu sestává ze dvou částí. Nejprve se snímek překryje obrázkem s vyznačenými výsledky detekce, následně je pak provedena konverze do barevného prostoru YUV, konkrétně reprezentace *I420.* Obě činnosti řeší jádro *rgbToI420Kernel()* jež je k nahlednutí v souboru *blend_convert.cu.*

Reprezentace *I420* využívá trojice bufferů, které však v paměti leží ihned za sebou. Ty z nich, jež uchovávají chromatické složky jsou však v obou osách zmenšeny na polovinu, neboť lidské oko je citlivější na jas než na barvu. Jádro proto v první fázi zjistí, zda má v daném pixelu ukládat chromatické složky. Následně provede načtení vstupní hodnoty v reprezentaci RGB, kterou volitelně aplikací operátoru *over* překryje rámečky. Jelikož knihovna *Cairo Graphics* reprezentuje obraz s průhledností v přednásobeném formátu (barevné složky násobeny průhledností), lze operátor *over* zjednodušit na výpočet vzorce $c_{out} = c_a + c_b \cdot (1 - \alpha_a)$, kde c_a reprezentuje obrazový bod v překryvné vrstvě, c_b v obraze videa a α_a průhlednost překryvu. Smíchaný obraz je pak převeden do jasové reprezentace aplikací výpočtů $Y = (0.229; 0.587; 0.114) \cdot (R; G; B)$, $U = 0.429 \cdot (B - Y)$ a $V = 0.877 \cdot (R - Y)$. Na konec je již výsledná reprezentace zapsána do příslušných pozic v paměti (chromatické pouze mají-li být uloženy) a odeslána do výstupní porce pipeline.

8.5 Integrace analýzy obrazu

K provedení samotné analýzy bylo opět využito frameworku *Darknet*. Ten však nebyl vhodný pro použití společně s analýzou neboť také vyžaduje použití CUDA jader. Jejich samotné volání je ovšem vždy blokující. Možným řešením by bylo přesunout analýzu do samostatného vlákna, avšak tato varianta není příliš vhodná, neboť jádra by byla spouštěna jen v případě, že je vlákno právě aktivní. Musela by tedy být řešena jistá forma synchronizace a přepínání kontextů vláken. Architektura CUDA ovšem nabízí použití takzvaných proudů. Ty jsou předány během volání jádrům a úlohy jsou tak při správném použití spouštěny paralelně.

V originální verzi frameworku, jež byla určena pro použití jak na klasickém procesoru, tak i grafickém akcelerátoru, nebylo proudů využíváno. Také alokoval zbytečně příliš mnoho nevyužité paměti a obsahoval (pro potřeby aplikace) množství nadbytečného kódu. Bylo tedy nutné kód přepracovat. Jednodušším přístupem, než originální verzi v celku upravovat a dohledávat potřebný kód, se ukázalo být postupně soubory původní knihovny přidávat do projektu na základě aktuální potřeby a vzájemných vazeb. Tak byly soubory zredukovány na nezbytné minimum. Další redukce připravila knihovnu o kód a alokace pro zpracování na CPU. Následovalo přidání podpory proudů a optimalizace paměťových přenosů. V některých místech docházelo ke zbytečnému kopírování mezi paměťovým prostorem procesoru a grafického akcelerátoru. Nakonec proběhlo doplnění synchronizace CUDA vláken do některých jader, což přineslo zlepšení výkonu (zřejmě u nesynchronizovaných jader docházelo ke kolizím a nevhodnému využívání cache). V současnosti je namigrována pouze podpora konvolučních, maxpooling a normalizačních vnitřních vrstev. Neměl by však být zásadní problém v případě potřeby zahrnout a upravit i další druhy.

Oběma výsledným proudům byl při volání inicializační procedury *cudaStreamCreateWithPriority()* předán příznak *cudaStreamNonBlocking*. Tím bylo dosaženo zařazení všech jader obou proudů do plánovače a jejich optimální spouštění s maximálním možným využitím grafického akcelerátoru. Bylo zamýšleno i využití priorit proudů, avšak ty nejsou v CUDA verze 6.5 podporovány. I při zadání libovolné hodnoty tak mají všechny proudy prioritu stejnou. Jestliže by tato funkcionalita byla v budoucnu nutná, nezbylo by než ji vyřešit nějakým alternativním přístupem. To by však obnášelo značné množství dalších experimentů.

9 Konfigurační webové rozhraní

Povinnou součástí kamery bylo dle zadání také webové rozhraní. Avšak bylo chápáno spíše jako podpůrná funkce, na jejíž využití by mělo být zařízení do budoucna připraveno. Z tohoto důvodu byly práce na něm vedeny spíše okrajově s ohledem na časový harmonogram. Při vývoji bylo rozhodnuto zaměřit se spíše na rozšiřitelnost než konkrétní funkcionalitu. Proto také poskytuje v současném stavu jen pár základních konfiguračních položek, jež se týkají analýzy a streamování. Je rozdělena na samostatný *backend* a *frontend.*

9.1 Backend

Jako webový server pro backend bylo zvoleno *Node.js* v kombinaci s frameworkem *Express.* Server pomocí modulu *routes* směřuje konkrétní URL k jejich obsluze do modulu *controllers.* Podporovány jsou následující cesty:

- */api/config* - čte (metoda GET) a ukládá (metoda POST) konfiguraci.
- */api/template* - poskytuje šablonu konfigurovatelných hodnot.
- */api/login* - zajišťuje přihlašování.
- */** - poskytuje soubory frontendu.

Veškeré umístění konfiguračních souborů je nastavitelné v souboru *config.js.* Bezpečnost a správa uživatelů není v tomto okamžiku hlouběji řešena (jedná se pouze o soubor s jménem a heslem v prostém textu) a bude nutné jej v případě pokračování projektu dopracovat. Sestavené soubory frontendu jsou umístěny v adresáři *dist.* Při změně konfigurace se spustí aplikace nebo skript (taktéž možno nastavit v souboru *config.js*), který zajistí propagaci příslušných nastavení do požadovaných míst, například zasláním signálu hlavní aplikaci.

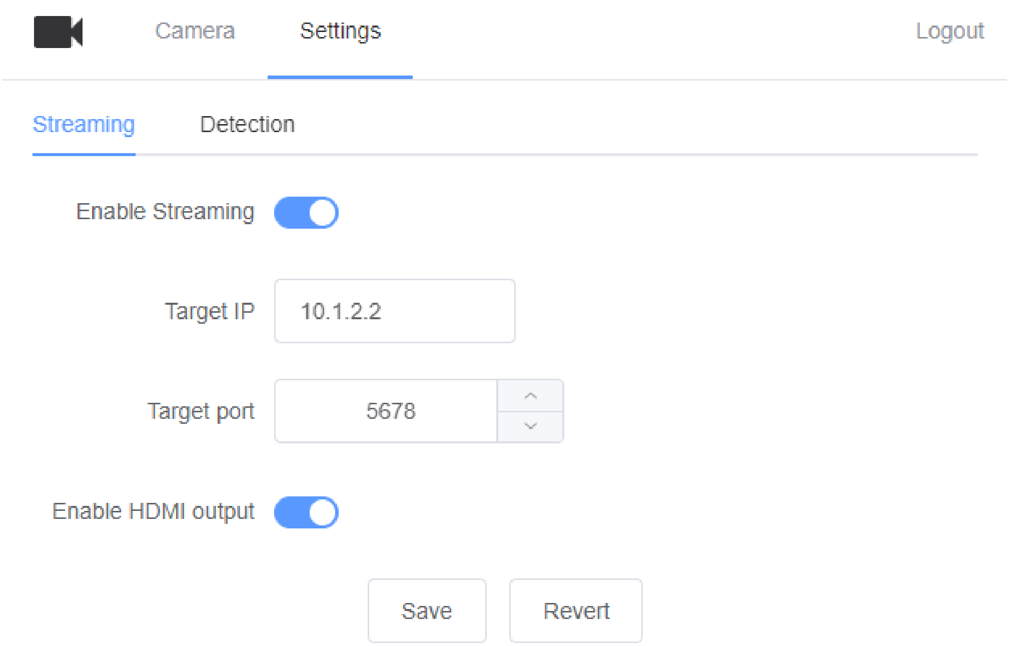
9.2 Frontend

Na doporučení zkušenějších kolegů v oblasti webových rozhraní bylo využito frameworku *Vue.js*[17].

Jedná se o rychlý, snadno použitelný a škálovatelný framework. Pro tvorbu uživatelského rozhraní je využita knihovna *Element-UI*[18], která je jednou z nejrozšířenějších (na portále GitHub dosahuje momentálně téměř 26000 tisíc „stars“), zejména proto, že je navržena přímo na míru frameworku *Vue.js* a obsahuje všechny nezbytné komponenty. Z uživatelského hlediska se pak jedná o jednostránkovou aplikaci, aby byla minimalizována komunikace s backendem na zařízení. Z tohoto důvodu byl pro přehlednost využit modul *Router*[19], který zajišťuje selektivní zobrazení komponent na základě voleb v rámci uživatelského rozhraní. Dále se využívá modulu *Auth*, který zajišťuje kontrolu přihlášení uživatele.

---

[17] `https://vuejs.org/` (dostupné 20. dubna 2018)
[18] `http://element.eleme.io/` (dostupné 20. dubna 2018)
[19] `https://router.vuejs.org/en/` (dostupné 20. dubna 2018)



Enable Streaming ☑

Target IP 10.1.2.2

Target port 5678

Enable HDMI output ☑

Obrázek 10: Náhled frontendu webového rozhraní

Frontend rozhraní je z hlediska struktury kódu rozdělen do šesti hlavních uživatelských komponent:

1. *App* - vždy zobrazená „top-level“ komponenta, která se skládá z komponent *Navbar* a *router-view* (komponenta vybraná modulem *Router*).
2. *Navbar* - navigační lišta umožňující pohyb v aplikaci pomocí modulů *Router* a *Auth*.
3. *Login* - přihlašovací obrazovka, ve spolupráci s modulem *Auth* ověřuje přihlášení a v závislosti na jeho úspěchu přesměrovává uživatele na komponentu *Camera* nebo zobrazuje chybovou notifikaci.
4. *Camera* - titulní stránka, v současnosti jen „placeholder“ pro budoucí využití (například náhled obrazu).
5. *Settings* - konfigurace kamery, hodnoty ke konfiguraci jsou definovány šablonou ve formátu *JSON*. Kategoricky jsou rozděleny do záložek reprezentovaných komponentou *Tab*. Dále zajišťuje ukládání a načítání konfigurovatelných hodnot.
6. *Tab* - zobrazuje konfigurovatelné hodnoty jedné kategorie. Každá položka je samostatnou drobnou komponentou se společným rozhraním, které akceptuje hodnotu a emituje novou hodnotu při její změně. Komponenta tab pak tuto hodnotu validuje a uchovává.

Pro konkrétní představu o vzhledu frontendu webového rozhraní je přiložen obrázek 10.

10 Zhodnocení výkonu zařízení

Aby bylo možné zhodnotit úspěšnost celé práce, je třeba zjistit, jak dobře zařízení zvládá řešení analytické úlohy realizované v rámci práce. Za tímto účelem byl do streamovací aplikace přidán výpis, který je možné zobrazit v ladícím režimu knihovny *gstreamer*. Konkrétně spuštěním aplikace s nastavením proměnné prostředí *GST_DEBUG* na úroveň 3 (výpis FIXME[20]) a více příkazem:

```
$ GST_DEBUG=3 streaming
```

Tím lze z aplikace získat každých 5 sekund údaje o rychlosti zpracování obrazových dat v následujícím formátu (pro přehlednost vynechávám hlavičku s informací o zdroji výpisu):

```
0:01:29.375300781 (...) Avg processing time: 0.077 sec (~12.93 FPS)
0:01:29.375945281 (...) Avg detection time: 0.202 sec (~4.95 FPS)
0:01:29.376514947 (...) Total frames per second (since last info): 8.53 (43
    frames per 5.039 sec)
```

Vzhledem k udávanému výkonu 290-365 GFLOPS a přibližné znalosti počtu operací neuronové sítě je možné odhadnout teoretickou horní hranici rychlosti analýzy na *14,5 - 18,25 snímků za sekundu*. Je však samozřejmé, že reálná hodnota bude nižší v důsledku další režie (například paměťových přenosů) a také proto, že v celkovém počtu operací nejsou zahrnuty operace potřebné ke zpracování streamovaného videa ve formátu 1080p (debayering, vyvážení bílé, ...).

Pro zhodnocení bylo zvoleno několik variant kombinací streamování, zobrazení prostřednictvím HDMI a analýzy. Ukázalo se, že zobrazení na výstupu HDMI a komprese videa s odesíláním po síti mají na rychlost zpracovávání zanedbatelný vliv (ve zobrazených statistikách prakticky nerozeznatelný). V případě samotného zpracování videa bez analýzy lze pozorovat téměř konstantní hodnotu *0.030* sekundy, což odpovídá přibližně *33 snímkům za sekundu*, avšak skrze zpracovávající část pipeline projde dle statistik reálně jen necelých *22 snímků za sekundu*. To lze však připisovat spíše metodice obsluhy vybírání obrazových dat ze snímače. V případě kombinace streamování s analýzou pak klesne rychlost zpracování toku obrazových dat přibližně na *13 snímků za sekundu* a paralelně probíhající analýza pak zpracuje přibližně necelých *5 snímků za sekundu*. Reálně skrze pipeline však projde okolo *9 snímků za sekundu*.

Pro úplnost je nutno dodat, že na ARM jádře procesoru (s využitím originální aplikace frameworku *darknet*) trvala analýza neuronovou sítí přibližně 13 sekund. Grafický akcelerátor tak zvládne tuto úlohu při současném zpracovávání videa přibližně *65*× rychleji.

---

[20] Ačkoli se dle dokumentace projektu *gstreamer* nejedná o korektní využití výpisu FIXME, využil jsem ji pro nízký počet zobrazených výpisů na této úrovni. Podstatná informace o rychlosti aplikace se tak neztratí v nezajímavém textu.

Bylo tak dosaženo uspokojivých výsledků, ke kterým byl projekt směřován, v následujících fázích vývoje se ale počítá s dalšími optimalizacemi. Ačkoli se snímkovací frekvence při zapnuté analýze může zdát nízká, pro potřeby dohledových systémů by měla být dostačující. Zejména z rozdílu mezi rychlostí jednotlivých úkonů a reálnou propustností pipeline je možné vyvodit, že vyššího výkonu bude pravděpodobně možno dosáhnout jejím dalším pokročilejším vylaďováním. Také je možné více zapracovat na vzájemné synchronizaci obou úloh tak, aby se co nejméně ovlivňovaly (zejména aby byly minimalizovány souběžné přístupy do paměti a paměťové přenosy vůbec) a aby bylo dosaženo ideálního poměru jejich rychlostí. Další místo pro optimalizaci pak skrývají samotná CUDA jádra obou úloh. Nejkrajnější variantou může být změna složitosti (a tím pravděpodobně i spolehlivosti) řešeného úkolu za účelem zvýšení výkonu zařízení. Všechny tyto optimalizace však budou vyžadovat hlubší analýzu a mnoho dalších experimentů v závislosti na konkrétních požadavcích dílčích projektů.

11 Závěr

Primárním záměrem této práce bylo navrhnout a sestavit elektronické zařízení schopné snímat obraz, zasílat jej dále a současně nad obrazem provádět úlohu z oblasti počítačového vidění a analýzy obrazu. Tato schopnost měla být prezentována na konkrétním příkladu, který by prokázal, že zařízení má dostatečný výkon tuto úlohu vykonávat. Jelikož se jednalo o projekt vypracovaný v rámci zaměstnání, bylo také důležité přinést co největší rozšiřitelnost a podporu technologií používaných v rámci ostatních projektů na pracovišti. Za tímto účelem vznikla po elektronické stránce prakticky kompletní kamera, která potenciálně pokrývá většinu případných využití (od záznamu na interní úložiště, přes činnost ve špatných světelných podmínkách, až po autonomní řízení dalších připojených zařízení).

V rámci práce se podařilo zkonstruovat plnohodnotný modulární a snadno rozšiřitelný koncept, který má dostatečný výkon pro provádění analýzy pomocí neuronové sítě čítající přes 19 miliard operací s plovoucí desetinnou čárkou a to při rychlosti 5 snímků za sekundu. Souběžně s tím dokáže komprimovat a přenášet video po síti Ethernet případně zobrazit na připojeném monitoru, rychlostí téměř 9 snímků za vteřinu. Optimalizací video pipeline však lze teoreticky dosáhnout rychlosti až 13 snímků za vteřinu. Nejedná se sice o vysoké, ale pro potřeby plánovaného využití přiměřeně dostatečné číslo. Spolehlivost prováděné detekce se pohybuje okolo 45% (jak přesností lokalizace, tak správností klasifikace). Zde je ovšem nutné zmínit, že síť byla vytrénována za účelem demonstrace výkonových schopností na generické datové sadě, jelikož pro žádnou konkrétní aplikaci nebylo zatím zařízení nabízeno a nejsou tedy k dispozici žádné přesné požadavky ani příslušná datová sada. Je tedy možno předpokládat dosažení vyšších přesností v případě nasazení produktu za konkrétním účelem.

Cíle uvedené v zadání v souladu s požadavky společností *CUTTER Systems spol. s r.o.* se zdají být naplněny, avšak v minimální nezbytné možné variantě. Celý projekt se ukázal být časově velmi náročným a doba strávená pracemi na projektu překročila 900 hodin pracovního i osobního času. Za přínos při realizaci této práce pak lze považovat výstupy dílčích vývojových prací, zejména pak odhalení problémů s jaderným modulem kamer v rámci systému *Linux for Tegra*, která může ostatním vývojářům citelně usnadnit řešení problémů s optickými snímači. Dalším přínosem je pak vytvoření univerzálního „odrazového můstku“ pro rychlý návrh odvozených kamerových zařízení, snadno rozšiřitelné streamovací a analytické aplikace a také vlastní rozšíření neuronové sítě *Tiny YOLO*, které při použití v rámci tohoto projektu podává lepší výsledky než originální síť. V rovině osobního rozvoje mi práce přinesla obohacení o nezanedbatelné množství nových poznatků v oblasti elektrotechniky a použité platformy.

Zařízení má také velký potenciál obohatit portfolio firmy. Pro další užití projektu bude nezbytné zprovoznit další periferie hardwaru, kterým nebyl v rámci této práce věnován dostatečný prostor, zejména kvůli časovým okolnostem zapříčiněným kritickými problémy během vývoje. Takovými periferiemi je například infračervený přísvit a senzor osvětlení na čelním panelu kamery. Z hlediska softwaru je pak řešení otevřeno dalším optimalizacím, a to mimo jiné také v závislosti na konkrétní analytické funkci, pro kterou bude zařízení využito. V případě dokončení dalších rozšíření tohoto zařízení a optimalizace analýzy pro konkrétní účel (a nad konkrétními daty) je velká pravděpodobnost jeho integrace do projektů navazujících na aktuální zakázky společnosti, jež se ve většině případů týkají zvyšování bezpečnosti pracovníků během jejich pohybu na rizikových pracovištích.

Literatura

[1] REDMON, Joseph. *YOLO: Real-Time Object Detection.* Joseph Redmon - Survival Strategies for the Robot Rebellion [online]. 2018 [cit. 2018-04-22]. Dostupné z: `https://pjreddie.com/darknet/yolo/`

[2] REDMON, Joseph, Santosh DIVVALA, Ross GIRSHICK a Ali FARHADI. *You Only Look Once: Unified, Real-Time Object Detection* [online]. 2016 [cit. 2018-04-22]. arXiv:1506.02640. Dostupné z: `https://arxiv.org/abs/1506.02640`

[3] REN, Shaoqing, Kaiming HE, Ross B. GIRSHICK a Jian SUN. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks* [online]. 2015 [cit. 2018-04-21]. arXiv:1506.01497. Dostupné z: `https://arxiv.org/abs/1506.01497`

[4] *COCO - Common Objects in Context* [online]. 2018 [cit. 2018-04-20]. Dostupné z: `http://cocodataset.org/`

[5] *Apalis TK1 Datasheet.* Toradex [online]. 2016, 23-Feb-2016 [cit. 2018-04-19]. Dostupné z: `https://docs.toradex.com/103129-apalis-tk1-datasheet.pdf`.

[6] *Apalis Computer Module: Carrier Board Design Guide.* Single Board Computers (SBCs), Computer on Modules, System on Modules [online]. [cit. 2018-04-19]. Dostupné z: `https://docs.toradex.com/101123-apalis-arm-carrier-board-design-guide.pdf`

[7] *Layout Design Guide.* Single Board Computers (SBCs), Computer on Modules, System on Modules [online]. [cit. 2018-04-19]. Dostupné z: `https://docs.toradex.com/102492-layout-design-guide.pdf`

[8] MORETTI, Gabe, ed. *Cadence Puts a Neural Network in a DSP.* ChipDesign Mag [online]. 2017, May 1st, 2017 [cit. 2018-04-19]. Dostupné z: `http://chipdesignmag.com/sld/blog/2017/05/01/cadence-puts-a-neural-network-in-a-dsp/`

[9] *Jetson Ecosystem.* NVIDIA Developer [online]. 2018 [cit. 2018-04-19]. Dostupné z: `https://developer.nvidia.com/embedded/community/ecosystem`

[10] OH, Nate, ed. *NVIDIA Gives Xavier Status Update & Announces TensorRT 3 at GTC China 2017 Keynote.* AnandTech [online]. 2017, September 26, 2017 [cit. 2018-04-19]. Dostupné z: `https://www.anandtech.com/show/11872/nvidia-xavier-status-update-and-tensorrt-3-announcement-at-gtc-china-2017-keynote`.

[11] *Press Release: Next generation SoC addition to Apalis computer module family.* Single Board Computers (SBCs), Computer on Modules, System on Modules [online]. 2015, March 6, 2015 [cit. 2018-04-19]. Dostupné z:
`https://www.toradex.com/news/next-generation-soc-apalis-computer-module-family`.

[12] *Apalis ARM Family.* Single Board Computers (SBCs), Computer on Modules, System on Modules [online]. [cit. 2018-04-19]. Dostupné z:
`https://www.toradex.com/computer-on-modules/apalis-arm-family`.

[13] *Variable SMP (4-PLUS-1^{TM}): A Multi-Core CPU Architecture for Low Power and High Performance.* NVIDIA [online]. 2012, 23. únor 2012 [cit. 2018-04-19]. Dostupné z:
`http://www.nvidia.com/content/PDF/tegra_white_papers/Variable-SMP-A-Multi-Core-CPU-Architecture-for-Low-Power-and-High-Performance.pdf`

[14] *Tegra: Tegra K1.* Wikipedia, the free encyclopedia [online]. 2018, 18 April 2018 [cit. 2018-04-19]. Dostupné z:
`https://en.wikipedia.org/wiki/Tegra#Tegra_K1`.

[15] *Mobile PCI Express Module.* Wikipedia, the free encyclopedia [online]. 2018, 16 April 2018 [cit. 2018-04-19]. Dostupné z:
`https://en.wikipedia.org/wiki/Mobile_PCI_Express_Module`.

[16] *NVIDIA Tegra K1 Computer on Module: Apalis TK1.* Single Board Computers (SBCs), Computer on Modules, System on Modules [online]. [cit. 2018-04-19]. Dostupné z:
`https://www.toradex.com/computer-on-modules/apalis-arm-family/nvidia-tegra-k1`.

[17] *M.2*, Wikipedia: the free encyclopedia [online]. 2018, 17 April 2018 [cit. 2018-04-20]. Dostupné z:
`https://en.wikipedia.org/wiki/M.2`

[18] *Ordering Information.* PCI-SIG [online]. [cit. 2018-04-19]. Dostupné z:
`http://pcisig.com/specifications/order-form`

[19] *Intel®Solid State Drive 540s Series (M.2): Product Specification*, Intel [online]. [cit. 2018-04-19]. Dostupné z:
`https://www.intel.com/content/dam/www/public/us/en/documents/product-specifications/ssd-540s-series-m2-spec.pdf`

[20] *FREEDOM KL25Z.* NXP Semiconductors [online]. [cit. 2018-04-19]. Dostupné z:
`https://www.nxp.com/downloads/en/schematics/FRDM-KL25Z_SCH_REV_E.pdf`

[21] *Circuit Converts PWM Fan Drive to Linear and Reduces Acoustic Noise: APPLICATION NOTE 3530*. Maxim [online]. [cit. 2018-04-19]. Dostupné z: `https://www.maximintegrated.com/en/app-notes/index.mvp/id/3530`

[22] *OV4686*. OmniVision [online]. [cit. 2018-04-19]. Dostupné z: `http://www.ovt.com/sensors/OV4686`

[23] *AR0330CM: 1/3-inch CMOS Digital Image Sensor*. ON Semiconductor [online]. [cit. 2018-04-19]. Dostupné z: `https://www.onsemi.com/pub/Collateral/AR0330CM-D.PDF`

[24] *Linux for Tegra R21.6*. NVIDIA Developer [online]. [cit. 2018-04-19]. Dostupné z: `https://developer.nvidia.com/linux-tegra-r216`

[25] *AR0330 on Apalis TK1*. Toradex Community [online]. [cit. 2018-04-19]. Dostupné z: `https://www.toradex.com/community/questions/20601/ar0330-on-apalis-tk1.html?childToView=20870#answer-20870`

[26] *Ar0330 on Jetson TK1*. NVIDIA Developer Forum [online]. [cit. 2018-04-19]. Dostupné z: `https://devtalk.nvidia.com/default/topic/853003/jetson-tk1/ar0330-on-jetson-tk1/`

[27] *Physical Hasher: The Bitmark Device*. GitHub [online]. [cit. 2018-04-19]. Dostupné z: `https://github.com/bitmark-inc/physical-hasher`

[28] HARRIS, Mark. *Prototyping Algorithms and Testing CUDA Kernels in MATLAB*. NVIDIA Developer Blog [online]. 2013, July 15, 2013 [cit. 2018-04-22]. Dostupné z: `https://devblogs.nvidia.com/prototyping-algorithms-and-testing-cuda-kernels-matlab/`

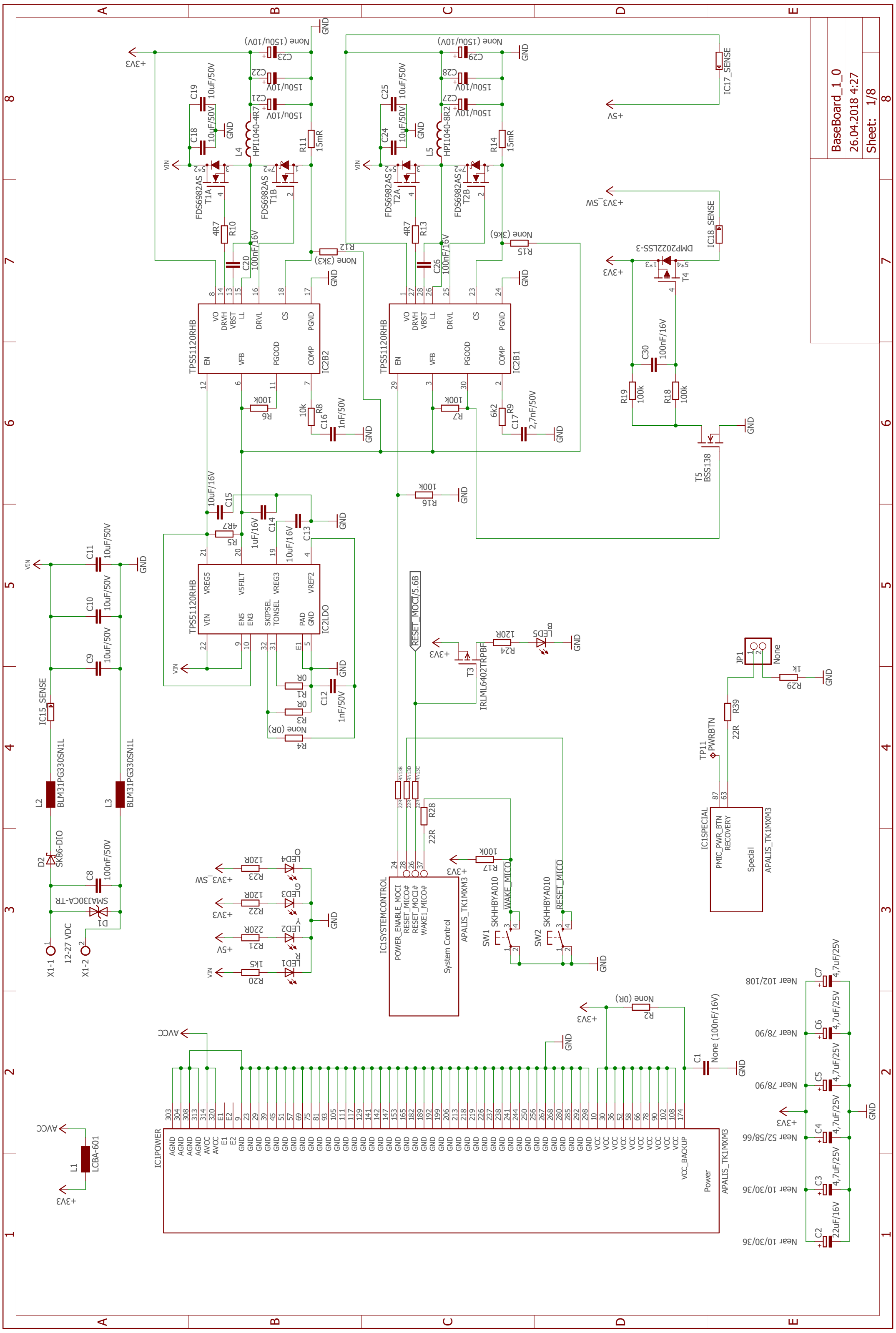
[29] GRENOBLE, Ryan. *Welcome To The Surveillance State: China's AI Cameras See All*. HuffPost [online]. 2017, 12/12/2017 [cit. 2018-04-25]. Dostupné z: `https://www.huffingtonpost.com/entry/china-surveillance-camera-big-brother_us_5a2ff4dfe4b01598ac484acc`

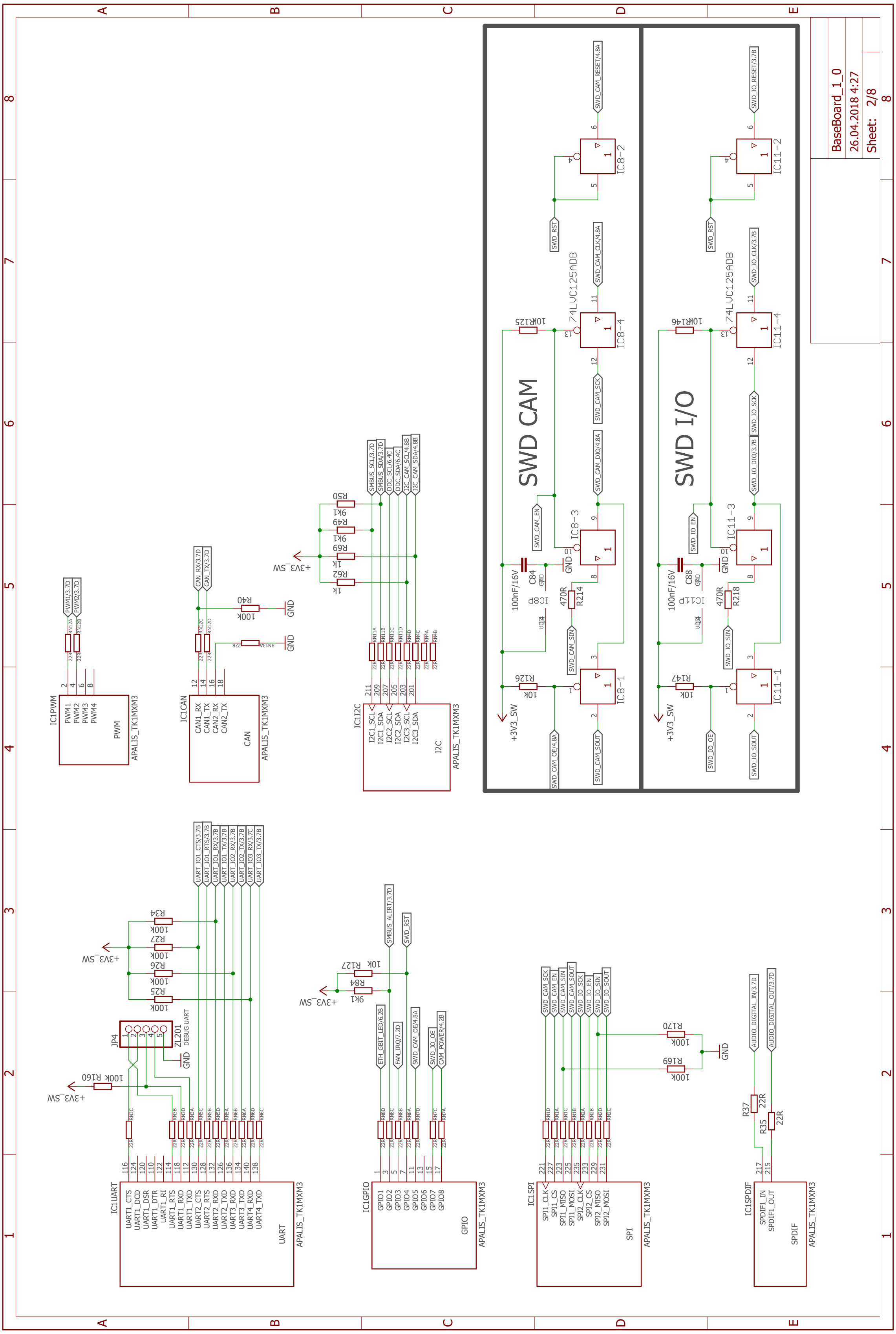
A Obsah CD s přílohami

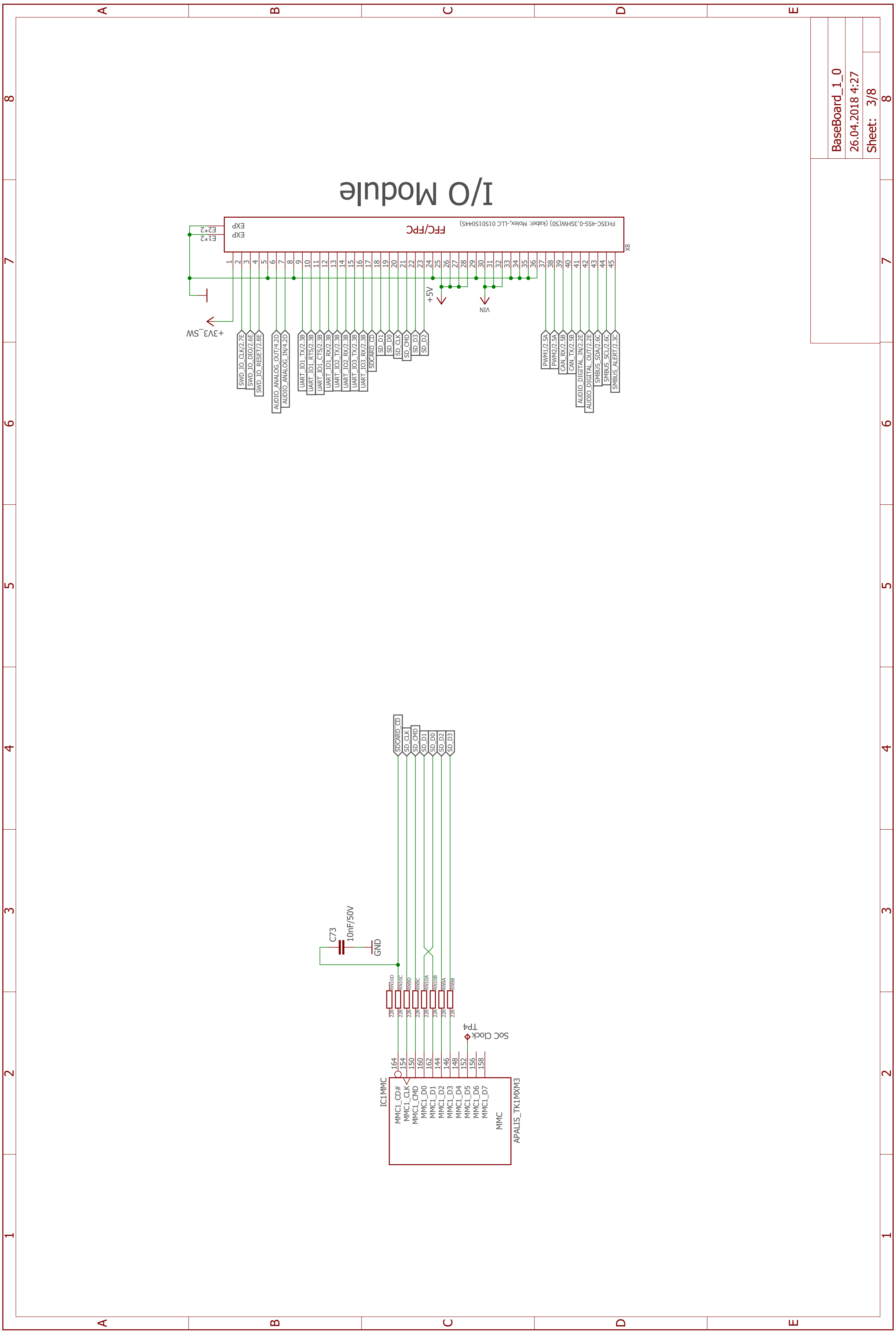
Přiložené CD obsahuje následující adresáře:

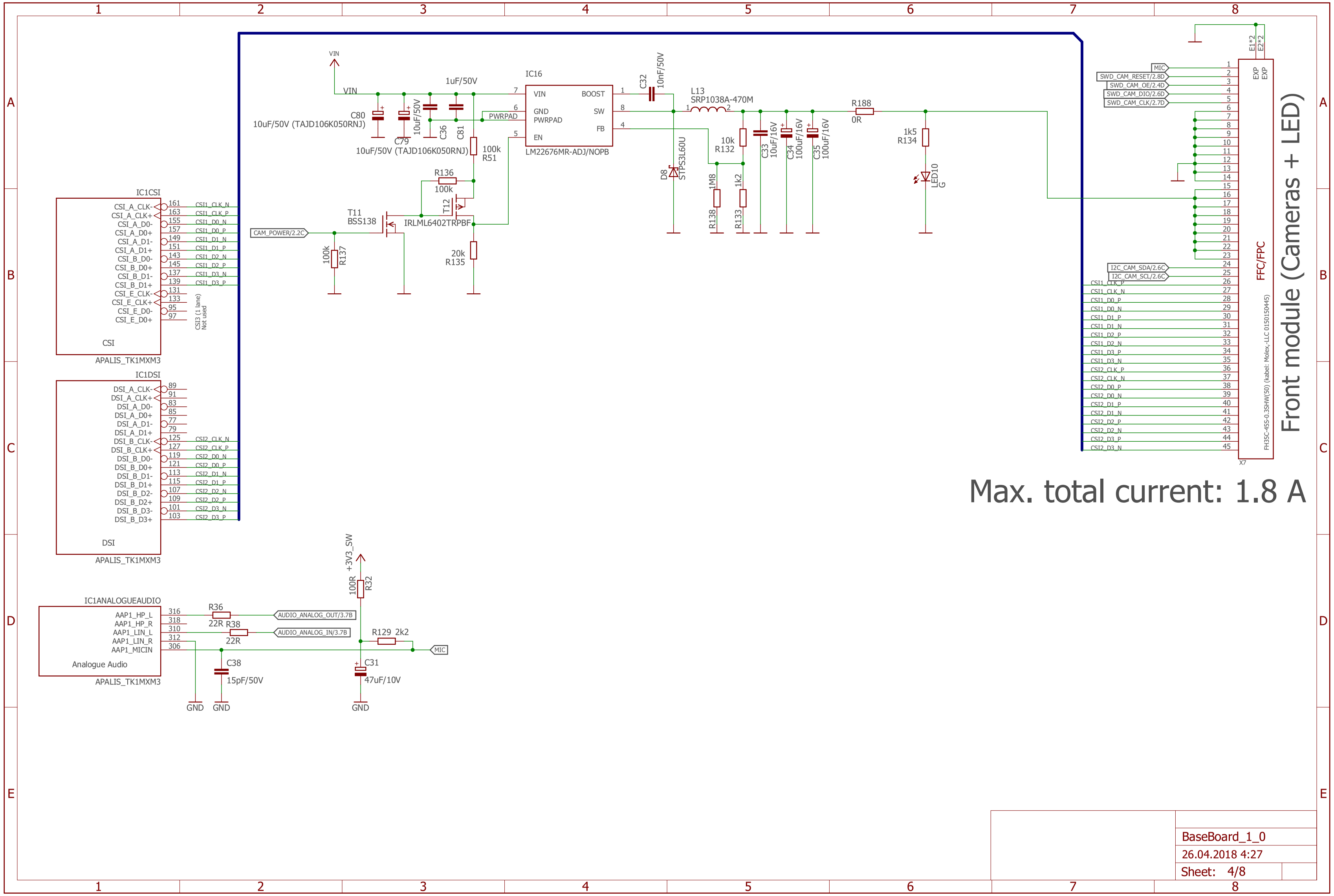
1. *Darknet* - framework Darknet pro Windows,
2. *Doc* - elektronická verze tohoto textu,
3. *Drawings* - elektronická schémata a desky plošných spojů,
4. *Firmware* - aplikace pro streamování videa a webová aplikace,
5. *Kernel* - patch soubor a konfigurační soubor pro sestavení jádra včetně binární podoby jádra a modifikovaných modulů,
6. *Print* - modely pro 3D tisk demonstračního rámu,
7. *Utils* - utility využité pro modifikaci datasetu COCO.

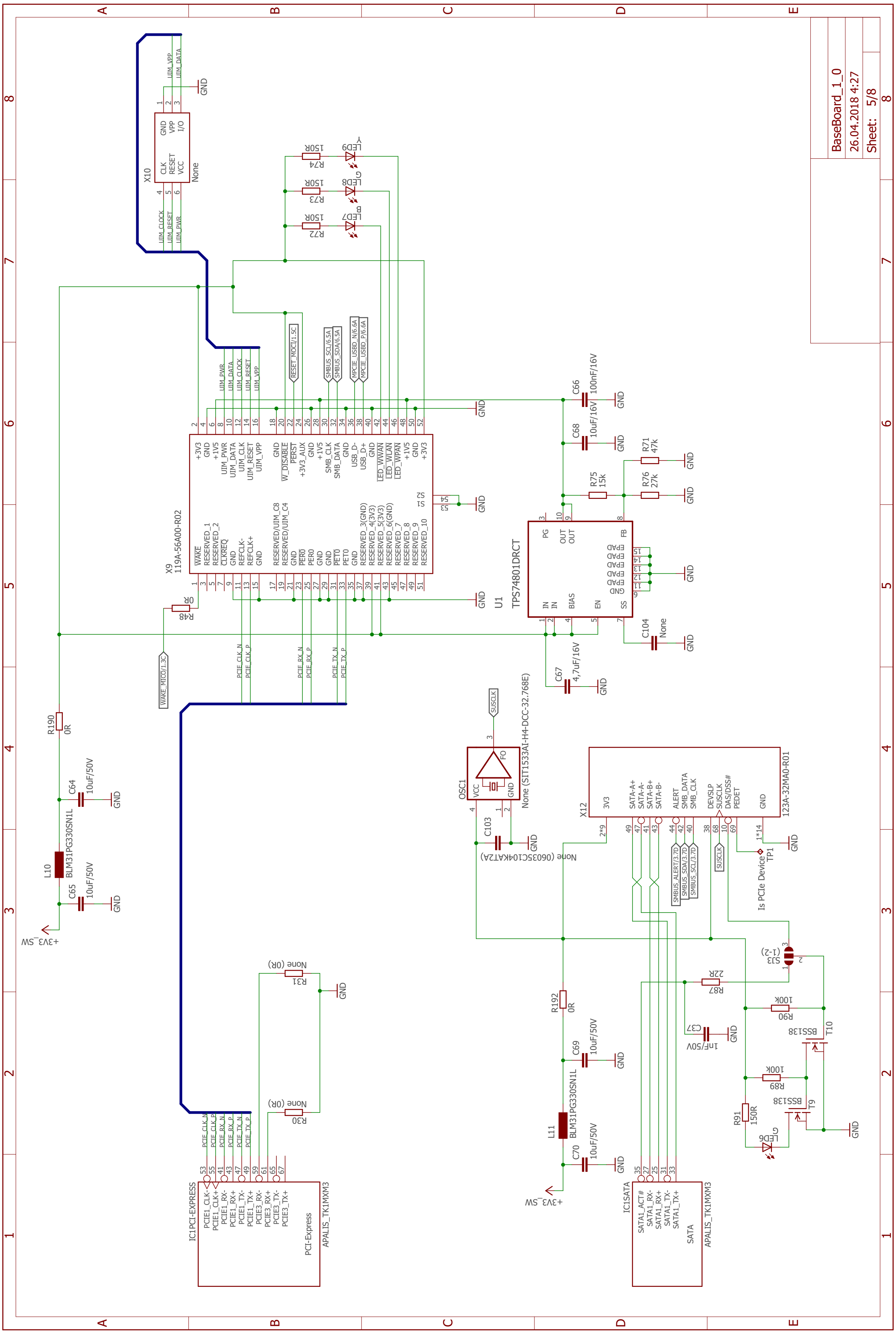
B Schéma hlavní desky

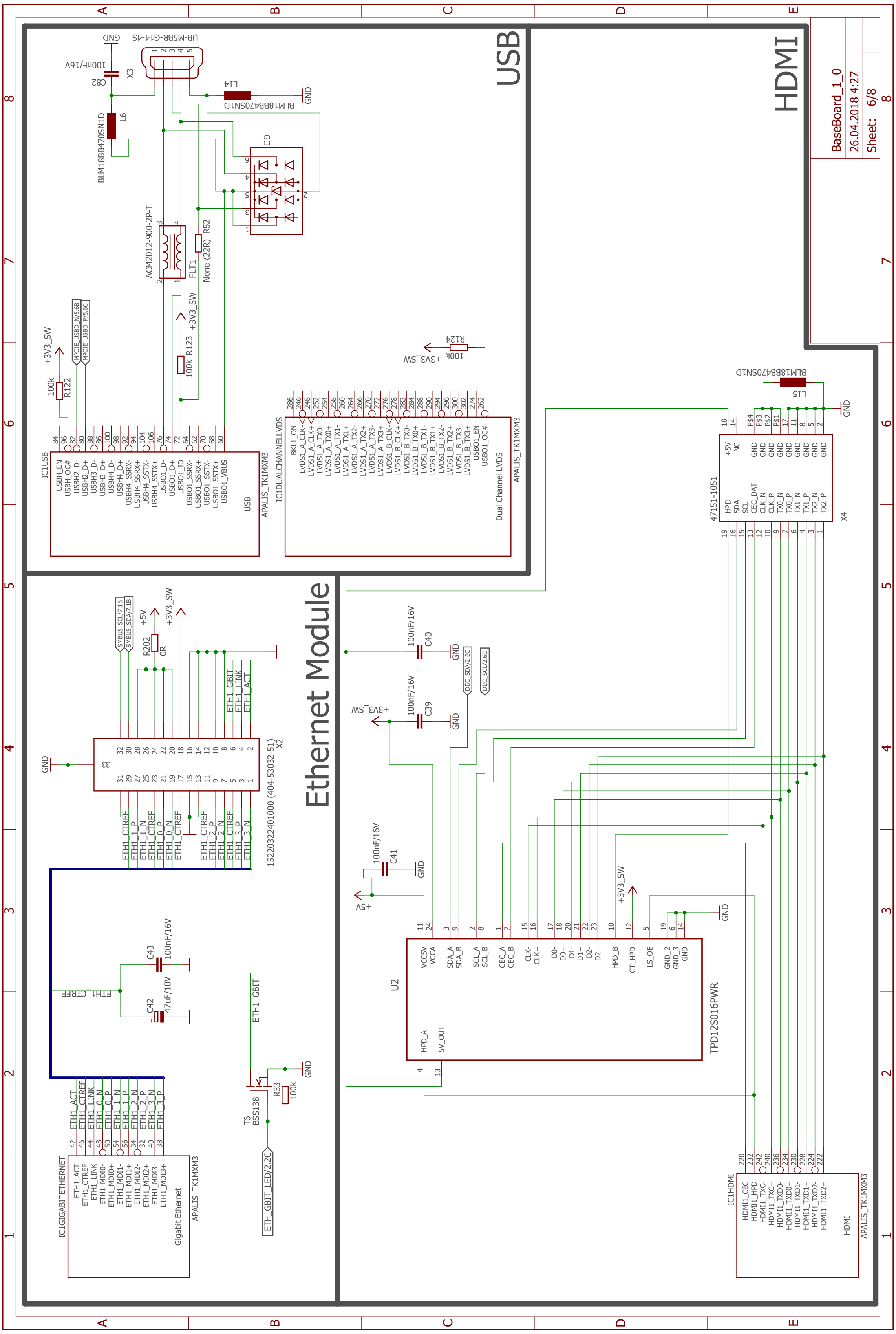


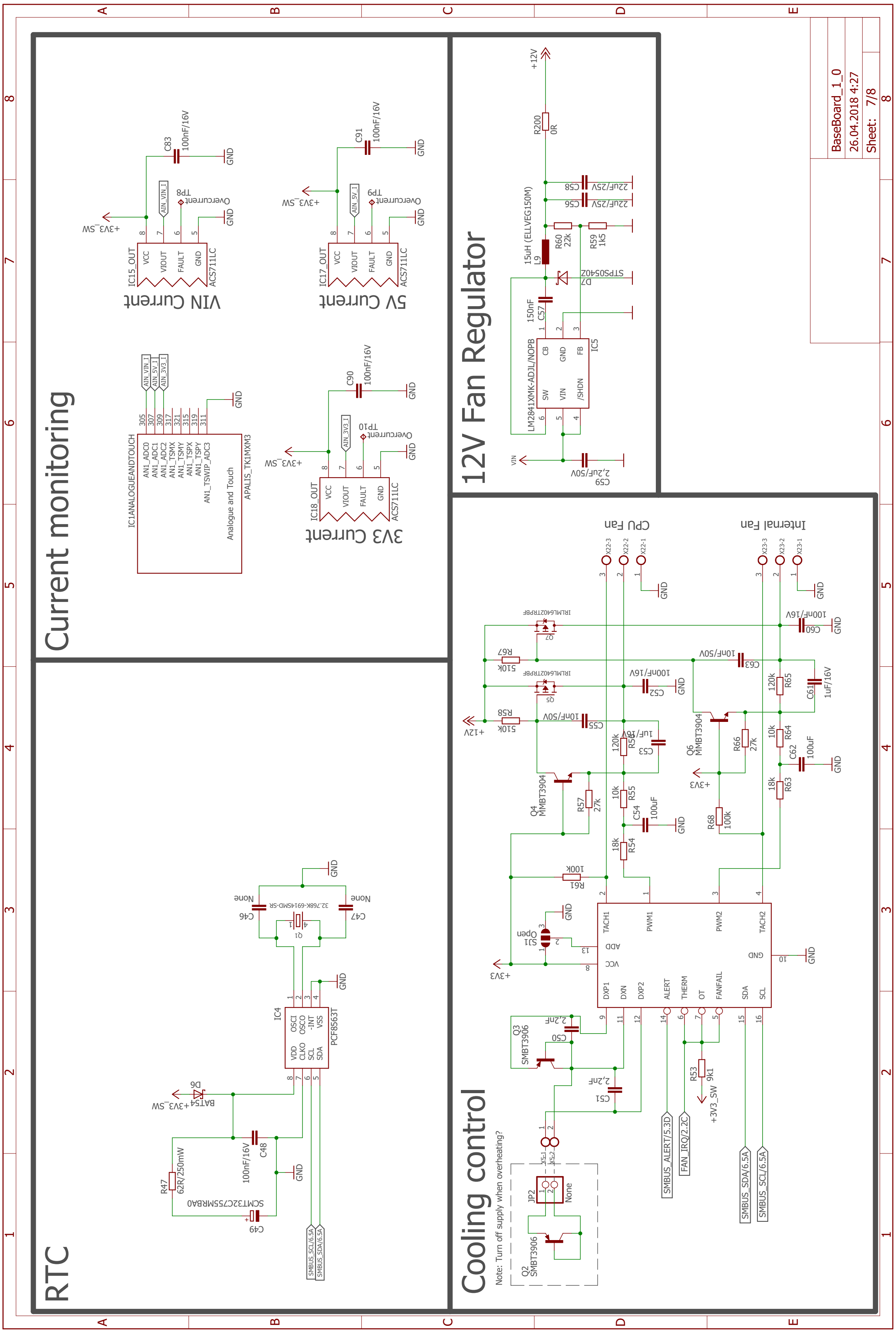

BaseBoard_1_0
26.04.2018 4:27
Sheet: 2/8
SWD CAM
SWD I/O
74LUC125ADB
IC8-1
IC8-2
IC8-3
IC8-4
IC11-1
IC11-2
IC11-3
IC11-4
+3V3_SW
GND
100nF/16V
C84
C88
IC8P
IC11P
470R
R214
R218
10K
R125
R126
R146
R147
SWD_RST
SWD_CAM_EN
SWD_IO_EN
SWD_CAM_SIN
SWD_IO_SIN
SWD_CAM_OE
SWD_IO_OE
SWD_CAM_SOUT
SWD_IO_SOUT
SWD_CAM_SCK
SWD_IO_SCK
SWD_CAM_DIO
SWD_IO_DIO
SWD_CAM_CLK
SWD_IO_CLK
SWD_CAM_RESET
SWD_IO_RESET
IC1PWM
PWM1
PWM2
PWM3
PWM4
PWM
APALIS_TK1MXM3
IC1CAN
CAN1_RX
CAN1_TX
CAN2_RX
CAN2_TX
CAN
R40
100K
IC1I2C
I2C1_SCL
I2C1_SDA
I2C2_SCL
I2C2_SDA
I2C3_SCL
I2C3_SDA
I2C
R50
R49
R69
R62
9k1
1k
SMBUS_SCL
SMBUS_SDA
DDC_SCL
DDC_SDA
I2C_CAM_SCL
I2C_CAM_SDA
IC1UART
UART1_CTS
UART1_DCD
UART1_DSR
UART1_DTR
UART1_RI
UART1_RTS
UART1_RXD
UART1_TXD
UART2_CTS
UART2_RTS
UART2_RXD
UART2_TXD
UART3_RXD
UART3_TXD
UART4_RXD
UART4_TXD
UART
JP4
ZL201
DEBUG UART
R160
R25
R26
R27
R34
UART_IO1_CTS
UART_IO1_RTS
UART_IO1_RX
UART_IO1_TX
UART_IO2_RX
UART_IO2_TX
UART_IO3_RX
UART_IO3_TX
IC1GPIO
GPIO1
GPIO2
GPIO3
GPIO4
GPIO5
GPIO6
GPIO7
GPIO8
GPIO
R84
R127
SMBUS_ALERT
ETH_GBIT_LED
FAN_IRQ
CAM_POWER
IC1SPI
SPI1_CLK
SPI1_CS
SPI1_MISO
SPI1_MOSI
SPI2_CLK
SPI2_CS
SPI2_MISO
SPI2_MOSI
SPI
R169
R170
IC1SPDIF
SPDIF1_IN
SPDIF1_OUT
SPDIF
R35
R37
22R
AUDIO_DIGITAL_IN
AUDIO_DIGITAL_OUT
A
B
C
D
E
1
2
3
4
5
6
7
8

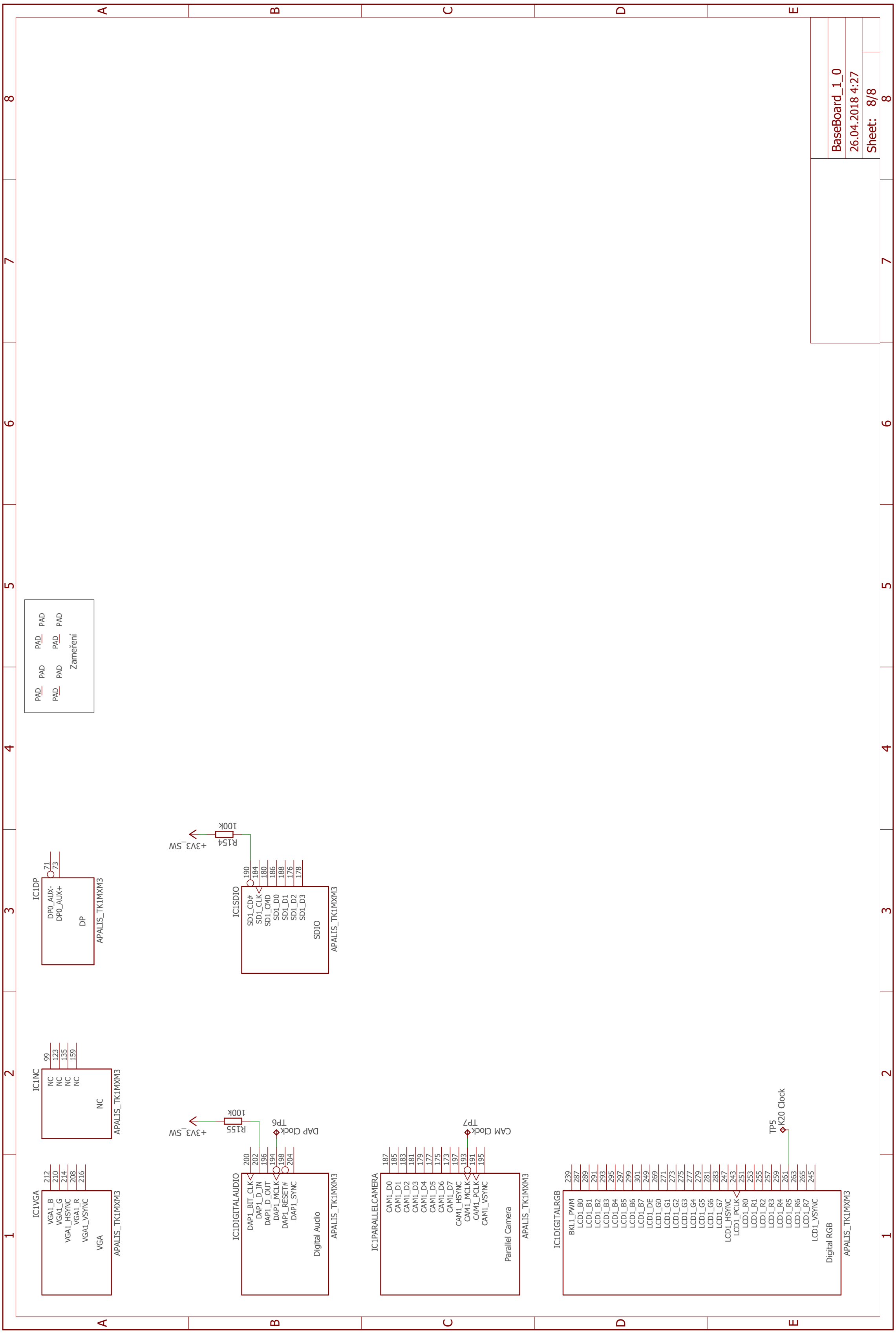

I/O Module
FFC/FPC
FH35C-45S-0.3SHW(50) (kabel: Molex,-LLC 0150150445)
X8
EXP
EXP
E1*2
E2*2
+3V3_SW
+5V
VIN
SWD_IO_CLK/2.7E
SWD_IO_DIO/2.6E
SWD_IO_RESET/2.8E
AUDIO_ANALOG_OUT/4.2D
AUDIO_ANALOG_IN/4.2D
UART_IO1_TX/2.3B
UART_IO1_RTS/2.3B
UART_IO1_CTS/2.3B
UART_IO1_RX/2.3B
UART_IO2_TX/2.3B
UART_IO2_RX/2.3B
UART_IO3_TX/2.3B
UART_IO3_RX/2.3B
SDCARD_CD
SD_D1
SD_D0
SD_CLK
SD_CMD
SD_D3
SD_D2
PWM1/2.5A
PWM2/2.5A
CAN_RX/2.5B
CAN_TX/2.5B
AUDIO_DIGITAL_IN/2.2E
AUDIO_DIGITAL_OUT/2.2E
SMBUS_SDA/2.6C
SMBUS_SCL/2.6C
SMBUS_ALERT/2.3C
C73
10nF/50V
GND
RN10D
RN10C
RN9D
RN9C
RN10A
RN10B
RN9A
RN9B
22R
TP4
SoC Clock
IC1MMC
MMC1_CD#
MMC1_CLK
MMC1_CMD
MMC1_D0
MMC1_D1
MMC1_D2
MMC1_D3
MMC1_D4
MMC1_D5
MMC1_D6
MMC1_D7
MMC
APALIS_TK1MXM3
164
154
150
160
162
144
146
148
152
156
158
BaseBoard_1_0
26.04.2018 4:27
Sheet: 3/8
A
B
C
D
E
1
2
3
4
5
6
7
8


Front module (Cameras + LED)
Max. total current: 1.8 A
FFC/FPC
FH35C-45S-0.3SHW(50) (kabel: Molex,-LLC 0150150445)
X7
EXP
E1*2
E2*2
MIC
SWD_CAM_RESET/2.8D
SWD_CAM_OE/2.4D
SWD_CAM_DIO/2.6D
SWD_CAM_CLK/2.7D
I2C_CAM_SDA/2.6C
I2C_CAM_SCL/2.6C
VIN
IC16
LM22676MR-ADJ/NOPB
BOOST
SW
FB
GND
PWRPAD
EN
C80
10uF/50V (TAJD106K050RNJ)
C79
10uF/50V (TAJD106K050RNJ)
C36
C81
1uF/50V
C32
10nF/50V
L13
SRP1038A-470M
D8
STPS3L60U
R132
10k
R138
1M8
R133
1k2
C33
10uF/16V
C34
100uF/16V
C35
100uF/16V
R188
0R
R134
1k5
LED10
G
R51
100k
R136
100k
T12
IRLML6402TRPBF
T11
BSS138
CAM_POWER/2.2C
R137
100k
R135
20k
IC1CSI
CSI
APALIS_TK1MXM3
CSI3 (1 lane)
Not used
IC1DSI
DSI
APALIS_TK1MXM3
IC1ANALOGUEAUDIO
Analogue Audio
APALIS_TK1MXM3
AAP1_HP_L
AAP1_HP_R
AAP1_LIN_L
AAP1_LIN_R
AAP1_MICIN
R36
22R
R38
22R
AUDIO_ANALOG_OUT/3.7B
AUDIO_ANALOG_IN/3.7B
R32
100R
+3V3_SW
R129
2k2
C38
15pF/50V
C31
47uF/10V
GND
BaseBoard_1_0
26.04.2018 4:27
Sheet: 4/8

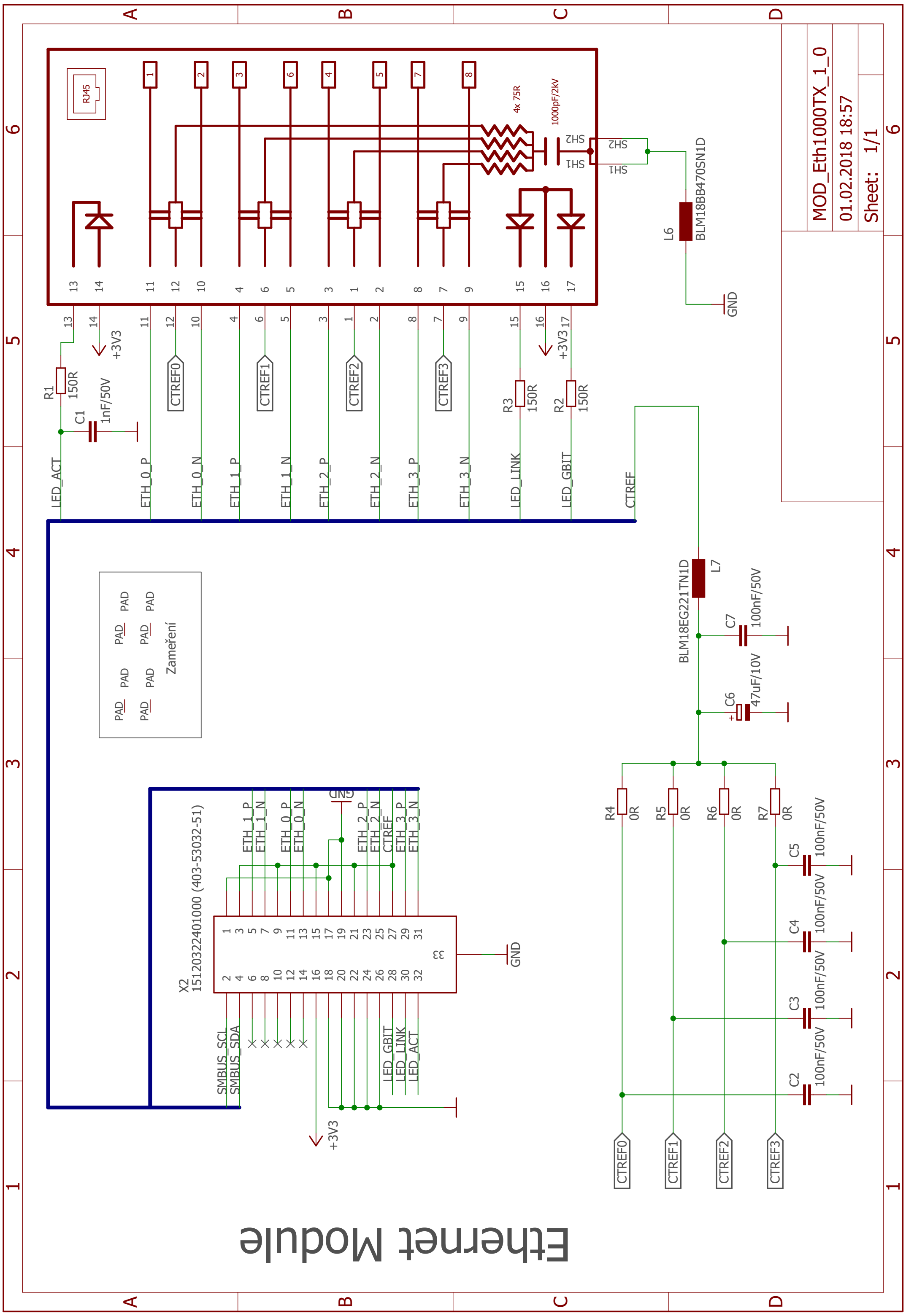

BaseBoard_1_0
26.04.2018 4:27
Sheet: 5/8
X9
119A-56A00-R02
X10
U1
TPS74801DRCT
OSC1
None (SIT1533AI-H4-DCC-32.768E)
X12
123A-32MA0-R01
IC1PCI-EXPRESS
PCI-Express
APALIS_TK1MXM3
IC1SATA
SATA
APALIS_TK1MXM3
L10
BLM31PG330SN1L
L11
BLM31PG330SN1L
+3V3_SW
GND
C64 10uF/50V
C65 10uF/50V
C66 100nF/16V
C67 4,7uF/16V
C68 10uF/16V
C69 10uF/50V
C70 10uF/50V
C103 None (0603SC104KAT2A)
C104 None
C37 1nF/50V
R190 0R
R192 0R
R48 0R
R30 None (0R)
R31 None (0R)
R71 47k
R75 15k
R76 27k
R72 150R
R73 150R
R74 150R
R87 22R
R89 100K
R90 100K
R91 150R
LED6
LED7
LED8
LED9
T9 BSS138
T10 BSS138
S13 (1-2)
TP1
Is PCIe Device
WAKE_MICO/1.3C
RESET_MOCI/1.5C
SMBUS_SCL/6.5A
SMBUS_SDA/6.5A
MPCIE_USBD_N/6.6A
MPCIE_USBD_P/6.6A
SMBUS_ALERT/3.7D
SMBUS_SDA/3.7D
SMBUS_SCL/3.7D
SUSCLK
UIM_PWR
UIM_DATA
UIM_CLOCK
UIM_RESET
UIM_VPP
PCIE_CLK_N
PCIE_CLK_P
PCIE_RX_N
PCIE_RX_P
PCIE_TX_N
PCIE_TX_P


USB
Ethernet Module
HDMI
BaseBoard_1_0
26.04.2018 4:27
Sheet: 6/8
IC1USB
APALIS_TK1MXM3
IC1DUALCHANNELLVDS
Dual Channel LVDS
IC1GIGABITETHERNET
Gigabit Ethernet
IC1HDMI
UB-M5BR-G14-4S
X3
C82
100nF/16V
L6
BLM18BB470SN1D
L14
L15
D9
ACM2012-900-2P-T
FLT1
R52
None (22R)
R122
R123
R124
100k
+3V3_SW
+5V
GND
X2
1522023240100 (404-53032-51)
R202
0R
ETH1_GBIT
ETH1_LINK
ETH1_ACT
ETH1_CTREF
C42
47uF/10V
C43
T6
BSS138
R33
ETH_GBIT_LED/2.2C
U2
TPD12S016PWR
C39
C40
C41
DDC_SDA/2.6C
DDC_SCL/2.6C
X4
47151-1051

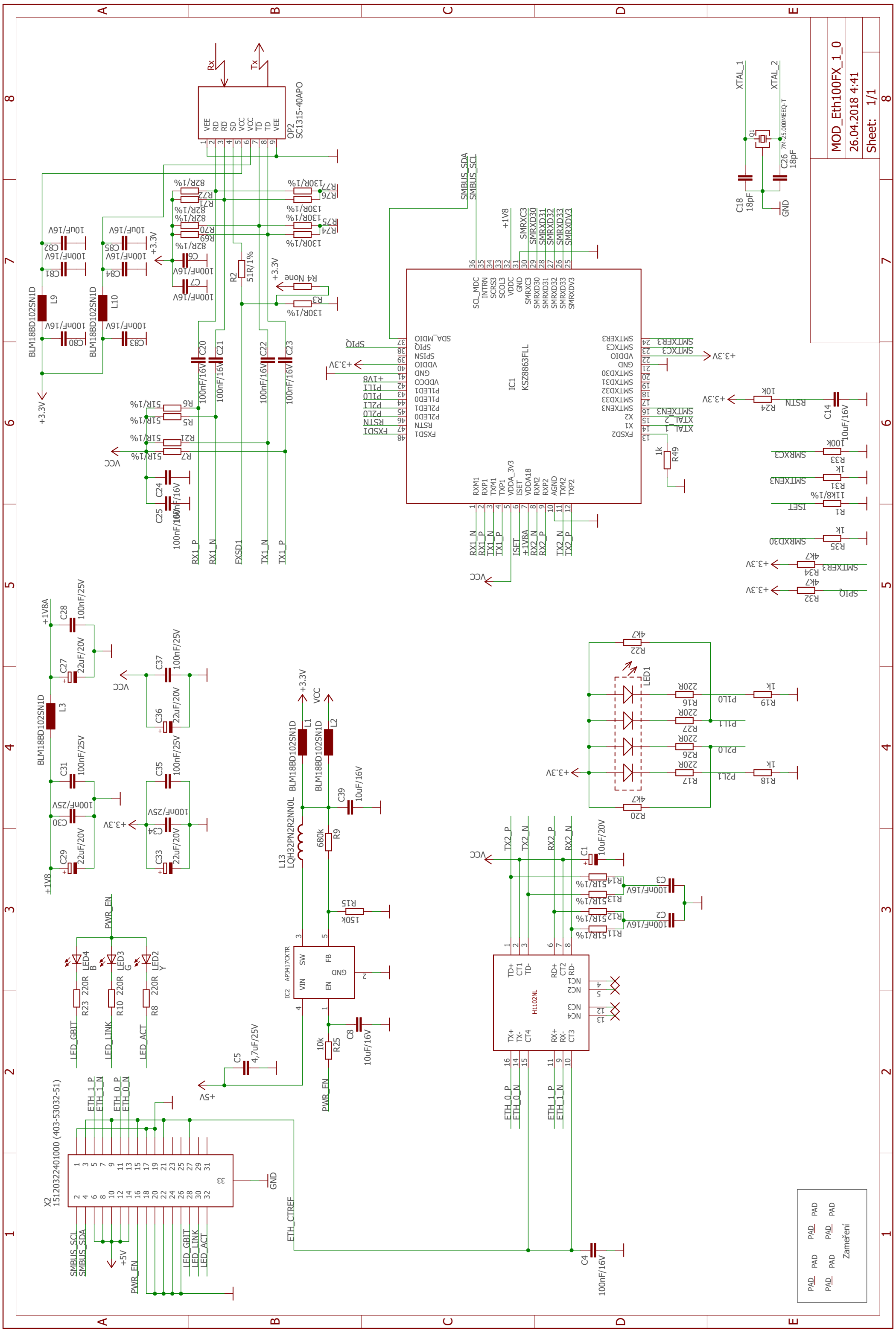

Current monitoring
VIN Current
IC15_OUT
VCC
VIOUT
FAULT
GND
ACS711LC
AIN_VIN_I
Overcurrent
TP8
+3V3_SW
C83
100nF/16V
GND
5V Current
IC17_OUT
AIN_5V_I
TP9
C91
100nF/16V
3V3 Current
IC18_OUT
AIN_3V3_I
TP10
C90
100nF/16V
IC1ANALOGUEANDTOUCH
AN1_ADC0
AN1_ADC1
AN1_ADC2
AN1_TSMX
AN1_TSMY
AN1_TSPX
AN1_TSPY
AN1_TSWIP_ADC3
Analogue and Touch
APALIS_TK1MXM3
12V Fan Regulator
+12V
R200
0R
C58
22uF/25V
C56
22uF/25V
R60
22k
R59
1k5
15uH (ELLVEG150M)
L9
D7
STPS0540Z
C57
150nF
LM2841XMK-ADJL/NOPB
CB
SW
VIN
GND
FB
/SHDN
IC5
VIN
C59
2,2uF/50V
RTC
IC4
PCF8563T
VDD
CLKO
SCL
SDA
OSCI
OSCO
-INT
VSS
Q1
32.768K-6914SMD-SR
C46
None
C47
None
D6
BAT54
+3V3_SW
R47
62R/250mW
C48
100nF/16V
C49
SCMT32C755MRBA0
SMBUS_SCL/6.5A
SMBUS_SDA/6.5A
Cooling control
Note: Turn off supply when overheating?
Q2
SMBT3906
JP2
None
Q3
SMBT3906
C50
2,2nF
C51
2,2nF
SJ1
Open
+3V3
TACH1
PWM1
PWM2
TACH2
ADD
VCC
GND
DXP1
DXN
DXP2
ALERT
THERM
OT
FANFAIL
SDA
SCL
R53
9k1
+3V3_SW
SMBUS_ALERT/5.3D
FAN_IRQ/2.2C
R61
100k
R54
18k
C54
100uF
R55
10k
R56
120k
C53
1uF/16V
R57
27k
Q4
MMBT3904
R58
510k
C55
10nF/50V
C52
100nF/16V
Q5
IRLML6402TRPBF
R67
510k
Q7
IRLML6402TRPBF
C63
10nF/50V
C60
100nF/16V
R68
100k
R66
27k
Q6
MMBT3904
+3V3
R63
18k
C62
100uF
R64
10k
R65
120k
C61
1uF/16V
CPU Fan
X22-1
X22-2
X22-3
Internal Fan
X23-1
X23-2
X23-3
BaseBoard_1_0
26.04.2018 4:27
Sheet: 7/8
A
B
C
D
E
1
2
3
4
5
6
7
8


BaseBoard_1_0
26.04.2018 4:27
Sheet: 8/8
IC1VGA
VGA1_B 212
VGA1_G 210
VGA1_HSYNC 214
VGA1_R 208
VGA1_VSYNC 216
VGA
APALIS_TK1MXM3
IC1NC
NC 99
NC 123
NC 135
NC 159
NC
APALIS_TK1MXM3
IC1DP
DP0_AUX- 71
DP0_AUX+ 73
DP
APALIS_TK1MXM3
PAD
Zaměření
IC1DIGITALAUDIO
DAP1_BIT_CLK 200
DAP1_D_IN 202
DAP1_D_OUT 196
DAP1_MCLK 194
DAP1_RESET# 198
DAP1_SYNC 204
Digital Audio
APALIS_TK1MXM3
R155
100K
+3V3_SW
TP6
DAP Clock
IC1SDIO
SD1_CD# 190
SD1_CLK 184
SD1_CMD 180
SD1_D0 186
SD1_D1 188
SD1_D2 176
SD1_D3 178
SDIO
APALIS_TK1MXM3
R154
100K
+3V3_SW
IC1PARALLELCAMERA
CAM1_D0 187
CAM1_D1 185
CAM1_D2 183
CAM1_D3 181
CAM1_D4 179
CAM1_D5 177
CAM1_D6 175
CAM1_D7 173
CAM1_HSYNC 197
CAM1_MCLK 193
CAM1_PCLK 191
CAM1_VSYNC 195
Parallel Camera
APALIS_TK1MXM3
TP7
CAM Clock
IC1DIGITALRGB
BKL1_PWM 239
LCD1_B0 287
LCD1_B1 289
LCD1_B2 291
LCD1_B3 293
LCD1_B4 295
LCD1_B5 297
LCD1_B6 299
LCD1_B7 301
LCD1_DE 249
LCD1_G0 269
LCD1_G1 271
LCD1_G2 273
LCD1_G3 275
LCD1_G4 277
LCD1_G5 279
LCD1_G6 281
LCD1_G7 283
LCD1_HSYNC 247
LCD1_PCLK 243
LCD1_R0 251
LCD1_R1 253
LCD1_R2 255
LCD1_R3 257
LCD1_R4 259
LCD1_R5 261
LCD1_R6 263
LCD1_R7 265
LCD1_VSYNC 245
Digital RGB
APALIS_TK1MXM3
TP5
K20 Clock
A B C D E
1 2 3 4 5 6 7 8

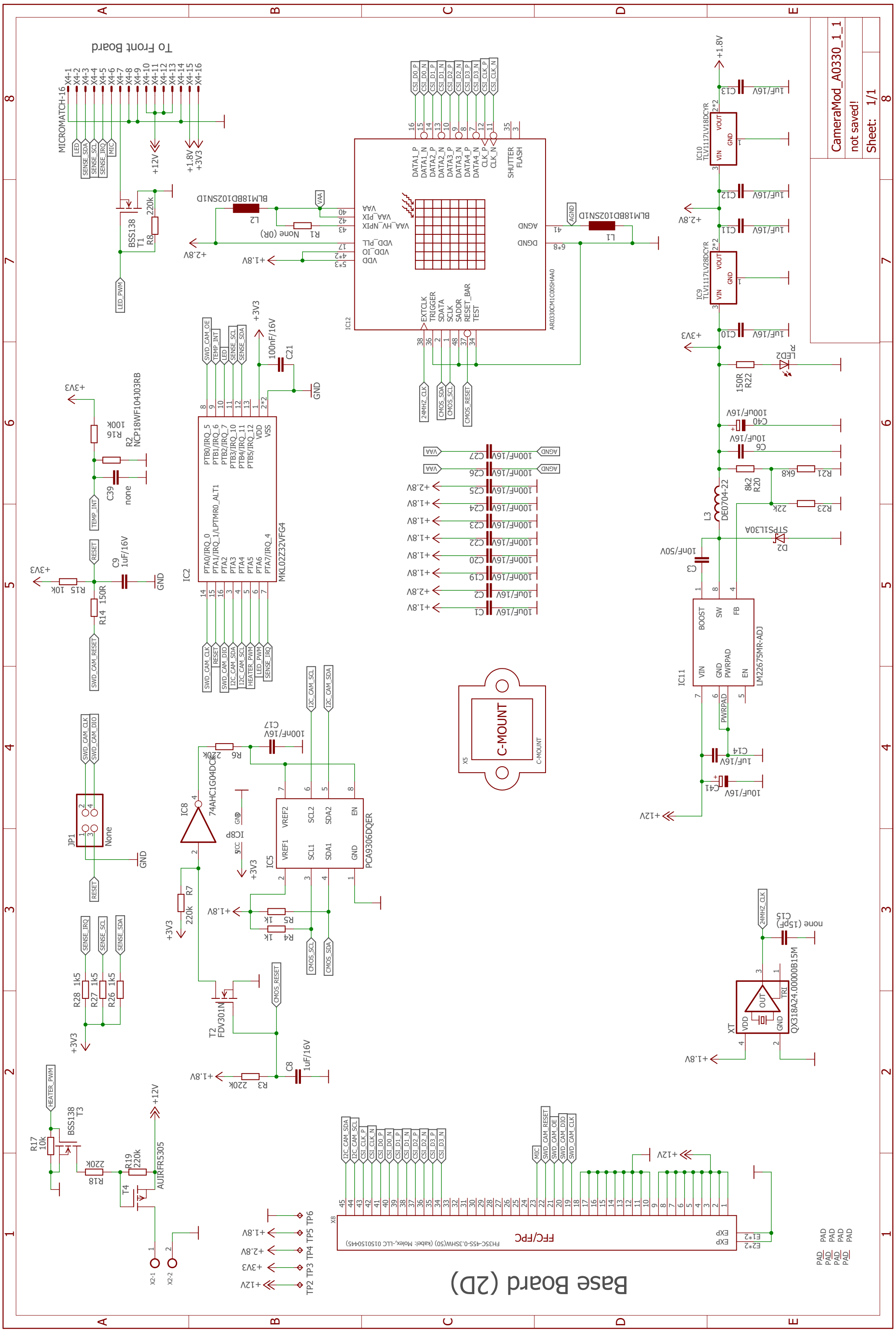
C Schéma modulu Ethernet BASE1000-TX



D Schéma modulu Ethernet BASE100-FX



E Schéma zadní desky modulu kamery



F Schéma přední desky modulu kamery